



alfamacursos.com.br

A decorative orange wave graphic that curves upwards from the bottom right corner of the page.

Alfama Cursos

Antônio Garcez
Fábio Garcez
Diretores Geral

Antônio Álvaro de Carvalho
Diretor Acadêmico

MATERIAL DIDÁTICO

Produção Técnica e Acadêmica

Patrícia Queiroz de Meneses
Coordenadora Geral

Patrícia Queiroz de Meneses
Coordenadora Pedagógica

Gabriella Caroline Teles Silva
Sabina Regina Conceição Santos
Revisão Textual

Rafael Rezende de Farias
Editoração

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/98.
É proibida a reprodução total ou parcial, por quaisquer meios, sem
autorização prévia, por escrito, da
ALFAMA CURSOS.

alfamacursos.com.br



Linguagem de Programação PHP

APRESENTAÇÃO DO CURSO

Hoje, o PHP é a quinta linguagem de programação mais usada no mundo, isso mostra a força que o PHP tem entre os desenvolvedores. O curso de Linguagem de Programação PHP permitirá ao aluno desenvolver sistemas web e sites. Durante o curso iremos conhecer toda a funcionalidade do PHP. Junto com HTML e com o MySQL iremos ver como armazenar dados no banco de dados e como recuperar esses dados.

APRESENTAÇÃO DO PROFESSOR

Tomaz Novaes Franco Pereira é graduado em Ciência da Computação pela Universidade Tiradentes (UNIT). Atualmente é desenvolvedor da Alfama Web, atuando na área de desenvolvimento web por mais 3 anos, desenvolvendo sistemas em várias linguagens de programação web, como por exemplo, PHP e JavaScript. Também é um dos responsáveis pelo desenvolvimento de sistemas de sucesso, como por exemplo, o Oráculo e o Construtor de Vendas, todos esses pertencentes à empresa Alfama Web.

EMENTA:

Introdução; Ferramentas auxiliares, Tag's HTML, Introdução ao PHP; Sintaxe do PHP; Comandos básicos do PHP, Tipos de dados; Array; Operadores; Estruturas de controle (Parte I); Estruturas de controle (Parte II); Estruturas de controle (Parte III); Funções; Classes e Objetos; GET, POST E FILES; PHP e MySQL (Parte I); PHP e MySQL (Parte II); PHP e MySQL (Parte III); Session; Cookies; Include, Require e Mail; Funções do PHP (Parte I); Funções do PHP (Parte II); PHP na Prática.

COMPETÊNCIAS:

- Relacionar os principais componentes envolvidos no ambiente da Internet, bem como compreender suas inter-relações.
- Reconhecer os princípios básicos da linguagem de programação PHP.
- Citar as principais vantagens de utilização das técnicas de programação, procedural e a orientada a objetos.

HABILIDADES:

- Demonstrar conhecimento sobre criação de aplicativos, com inclusão de formulários, validação de dados, integração com o Sistema de Arquivos, Banco de Dados externos e autenticação de usuários.
- Utilizar as práticas recomendadas para programação e segurança dos aplicativos.
- Identificar as técnicas de programação, como a Procedural e a Orientada a Objetos, bem como reconhecer suas principais vantagens.

PÚBLICO-ALVO:

Estudantes de informática, técnicos de diversas áreas, empresas e repartições públicas.

Sumário

1 - INTRODUÇÃO	2
1.1 – LINGUAGEM DE PROGRAMAÇÃO	2
1.2 – HTML.....	2
1.3 – PHP.....	3
1.4 – MYSQL	4
1.5 - APACHE	5
1.6 - EXERCÍCIO PRÁTICO	Erro! Indicador não definido.
2 – FERRAMENTAS AUXILIARES.....	8
2.1 – WAMPSEVER	8
2.1.1 – INSTALANDO O WAMPSEVER.....	8
2.1.2 – CONFIGURANDO O WAMPSEVER	12
2.2 – NOTEPAD++	15
2.3 – PHPMYADMIN	16
2.4 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
3 – TAG’S HTML.....	18
3.1 – TAG’S <HTML>	18
3.2 – TAG’S <HEAD>	18
3.3 – TAG’S <TITLE>	18
3.4 – TAG’S <BODY>	18
3.5 – ESTRUTURA BÁSICA DO HTML.....	19
3.6 – FORMULÁRIO HTML	19
3.6.1 – <INPUT>.....	20
3.6.2 – Radio	20
3.6.3 – Checkbox.....	21
3.6.4 – Select.....	21
3.6.5 – Textarea.....	22
3.6.6 – Botão	22
3.7 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
4 – INTRODUÇÃO AO PHP	24
4.1 – HISTORIA DO PHP	Erro! Indicador não definido.
4.2 – O QUE PODE SER FEITO COM O PHP?.....	24

4.3 – CARACTERISTICAS DO PHP	25
4.4 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
5 – SINTAXE DO PHP.....	28
5.1 – DELIMITANDO O CÓDIGO PHP	28
5.2 – SEPARADOR DE INSTRUÇÕES	29
5.3 – COMENTÁRIOS	29
5.4 – VARIÁVEIS	30
5.5 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
6 – COMANDOS BÁSICOS DO PHP	32
6.1 – ECHO	32
6.2 – PRINT	32
6.3 – CONCATENAÇÃO.....	33
6.4 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
7 – TIPOS DE DADOS.....	36
7.1 – INTEIROS (INTEGER E LONG)	36
7.2 – REAIS (FLOAT E DOUBLE)	36
7.3 – CARACTERES ALFANUMÉRICOS (STRING).....	37
7.4 – LÓGICOS (BOOLEANO).....	37
7.5 – ARRAYS.....	37
7.6 – OBJETOS.....	38
7.7 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
8 - ARRAY	41
8.1 – PRINT_R.....	42
8.2 – COUNT	42
8.3 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
9 - OPERADORES	44
9.1 – ARITMÉTICOS	44
9.2 – ATRIBUIÇÃO	44
9.3 – COMPARAÇÃO.....	45
9.4 – LÓGICO.....	46
9.5 – INCREMENTO E DECREMENTO	47
9.6 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
10 – ESTRUTURAS DE CONTROLE (PARTE I).....	49
10.1 – BLOCOS	49

10.2 – IF	49
10.3 – ELSE	50
10.4 – ELSEIF	51
10.5 – SWITCH	52
10.6 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
11 - ESTRUTURAS DE CONTROLE (PARTE II)	56
11.1 – WHILE	56
11.2 – WHILE	56
11.3 – FOR.....	57
11.4 – BREAK	58
11.5 – CONTINUE	59
11.6 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
12 - ESTRUTURAS DE CONTROLE (PARTE III).....	62
12.1 – EXEMPLO DE UTILIZAÇÃO DO IF	62
12.2 – EXEMPLO DE UTILIZAÇÃO DO SWITCH	63
12.3 – EXEMPLO DE UTILIZAÇÃO DO WHILE, DO...WHILE E FOR	65
12.4 – EXEMPLO DE UTILIZAÇÃO DO FOR	66
12.5 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
13 - FUNÇÕES	69
13.1 – DEFINIÇÃO DE FUNÇÕES	69
13.2 – RETORNO	69
13.3 – PARÂMETROS	69
13.4 – PARÂMETROS COM VALORES PRÉ-DEFINIDOS (DEFAULT)	70
13.5 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
14 – CLASSES E OBJETOS	73
14.1 – CLASSES.....	73
14.2 – OBJETOS.....	74
14.3 – SUBCLASSES	74
14.4 – CONSTRUTORES.....	75
14.5 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
15 – GET, POST E FILE	77
15.1 – GET	77
15.2 – POST	78
15.3 – FILES	79

15.4 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
16 – PHP e MySQL (PARTE I)	82
16.1 – INTRODUÇÃO AO MYSQL	82
16.2 – FUNÇÕES DO PHP PARA O MYSQL	82
16.2.1 – MYSQL_CONNECT	82
16.2.2 – MYSQL_SELECTDB.....	82
16.2.3 – MYSQL_QUERY	82
16.2.4 – EXEMPLO	83
16.3 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
17 – PHP e MySQL (PARTE II).....	85
17.1 – EXIBIÇÃO (mysql_fetch_assoc).....	85
17.2 – FUNÇÕES DO PHP PARA O MYSQL	85
17.2.1 – MYSQL_NUM_ROWS	85
17.2.2 – MYSQL_ERROR.....	86
17.2.3 – EXEMPLO	86
17.3 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
18 – PHP e MySQL (PARTE III)	88
18.1 – INSERÇÃO.....	88
18.2 – ATUALIZAÇÃO	88
18.3 – EXCLUSÃO	89
18.4 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
19 – SESSION.....	91
19.1 – INICIANDO UMA SESSION	91
19.2 – FINALIZANDO UMA SESSION	92
19.3 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
20 – COOKIES	94
20.1 – INICIANDO UMA COOKIE	94
20.2 – FINALIZANDO UMA COOKIE	95
20.3 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
21 – INCLUDE, REQUIRE, MAIL	97
21.1 – INCLUDE	97
21.2 – REQUIRE	97
21.3 – MAIL.....	98
21.4 - EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.

22	– FUNÇÕES DO PHP (PARTE I).....	101
22.1	– DATE	101
22.2	– NL2BR	105
22.3	– JOIN	105
22.4	– EXPLODE.....	106
22.5	- EXERCÍCIO PROPOSTO.....	Erro! Indicador não definido.
23	– FUNÇÕES DO PHP (PARTE II).....	108
23.1	– TRIM.....	108
23.2	– STRTOLOWER	108
23.3	– STRTOUPPER.....	109
23.4	– UNSET	109
23.5	– STR_REPLACE.....	110
22.6	- EXERCÍCIO PROPOSTO.....	110
24	– PHP NA PRÁTICA.....	112
24.1	– CLASSE DE CONTATO.....	112
24.2	– FORMULÁRIOS	113
24.3	– LISTAGEM.....	117
24.4	– LOGIN	119

Capítulo 1

Introdução

1 - INTRODUÇÃO

1.1 - LINGUAGEM DE PROGRAMAÇÃO

É uma padronização de método para passar instruções ao computador. Permite ao programador especificar precisamente quais dados o computador vai atuar, armazenar ou transmitir.

Podemos imaginar o computador como uma super calculadora, capaz de fazer cálculos muito mais rápido que nós, mas para isso devemos dizer para o computador o que deve ser calculado e como deve ser calculado. A função das linguagens de programação é exatamente essa, ou seja, servir de um meio de comunicação entre computadores e humanos.



1.2 - HTML

HyperText Markup Language ou Linguagem de Marcação HiperTexto, uma linguagem simples composta de marcações de formatação e diagramação de hipertexto/hipermídia (informações em texto, imagens, sons e ações ligadas umas às outras de uma forma complexa e não sequencial através de chaves relacionadas).

É a linguagem da WWW (World Wide Web), justamente por essa capacidade de formatação e diagramação de hipertexto/hipermídia. Atualmente existem muitas outras linguagens

utilizadas concorrentemente com o HTML, mas a base da WWW ainda é, de longe, o HTML, que é interpretada por todos os navegadores (browsers) disponíveis.

O documento em HTML é um arquivo texto comum (ASCII) que pode ser escrito através de qualquer editor de textos comum e tem a terminação .htm ou .html.

Como é uma linguagem interpretada, a partir do momento em que se salvou o arquivo HTML, o mesmo já pode ser aberto por qualquer navegador, que é o único software que pode determinar se o documento foi escrito dentro dos padrões da linguagem.



1.3 - PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. A diferença de PHP com relação a linguagens semelhantes ao JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas HTML puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código-fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

O que diferencia PHP de um script CGI escrito em C ou Perl é que o código PHP fica embutido no próprio HTML, enquanto no outro caso é necessário que o script CGI gere todo o código HTML, ou ler de um outro arquivo.

```

27 <!-- begin middle table -->
28 <table width="100%" border="0" cellspacing="0" cellpadding="0">
29   <tr>
30     <td>
31       <table width="100%" border="0" cellspacing="0" cellpadding="0">
32         <tr>
33           <!-- begin left column -->
34           <td width="18%" valign="top"><?php mosLoadModules ( 'left' ); ?>
35           <?php mosLoadModules ( 'user1' ); ?></td>
36           <!-- begin left column -->
37           <!-- begin center column -->
38           <td valign="top"><?php mosMainBody(); ?>
39                               </td>
40           <!-- end center column -->
41           <!-- begin right column -->
42           <td width="18%" valign="top"><?php mosLoadModules ( 'right' ); ?>
43           <?php mosLoadModules ( 'user2' ); ?></td>
44           <!-- end right column -->
45         </tr>
46       </table>
47     </td>
48   </tr>
49 </table>

```

1.4 - MYSQL

É um Sistema de Gerenciamento de Banco de Dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo.

Entre os usuários do banco de dados MySQL estão: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S Army, US. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems, Google CanaVialis S.A. e outros.



1.5 - APACHE

Ao acessar qualquer site, há um servidor por trás daquele endereço responsável por disponibilizar as páginas e todos os demais recursos que você pode acessar. Assim, quando você envia um e-mail através de um formulário, coloca uma mensagem em um fórum de discussão, faz uma compra on-line, etc., um servidor Web (ou um conjunto de servidores) é responsável por processar todas essas informações.

Sendo mais claro, um servidor Web é um computador que processa solicitações HTTP (*Hyper-Text Transfer Protocol*), o protocolo padrão da Web. Quando você usa um navegador de internet para acessar um site, este faz as solicitações devidas ao servidor Web do site através de HTTP e então recebe o conteúdo correspondente. No caso do Apache, ele não só executa o HTTP, como outros protocolos, tais como o HTTPS (O HTTP combinado com a camada de segurança SSL - *Secure Socket Layer*), o FTP (*File Transfer Protocol*), entre outros.

Como servidor Web, o Apache é o mais conhecido e usado. Os motivos incluem sua excelente performance, segurança, compatibilidade com diversas plataformas e todos os seus recursos.



1.6 - EXERCÍCIOS PROPOSTOS

1. O que é Linguagem de Programação?

2. Fale sobre HTML, MySQL e Apache.

Capítulo 2

Ferramentas Auxiliares

2 - FERRAMENTAS AUXILIARES

2.1 - WAMPSEVER

WampServer é um software para Windows de desenvolvimento Web. Ele permite que você crie aplicações Web com Apache, PHP e banco de dados MySQL.

- **W**indows
- **A**pache
- **M**ySQL
- **P**HP



2.1.1 - INSTALANDO O WAMPSEVER

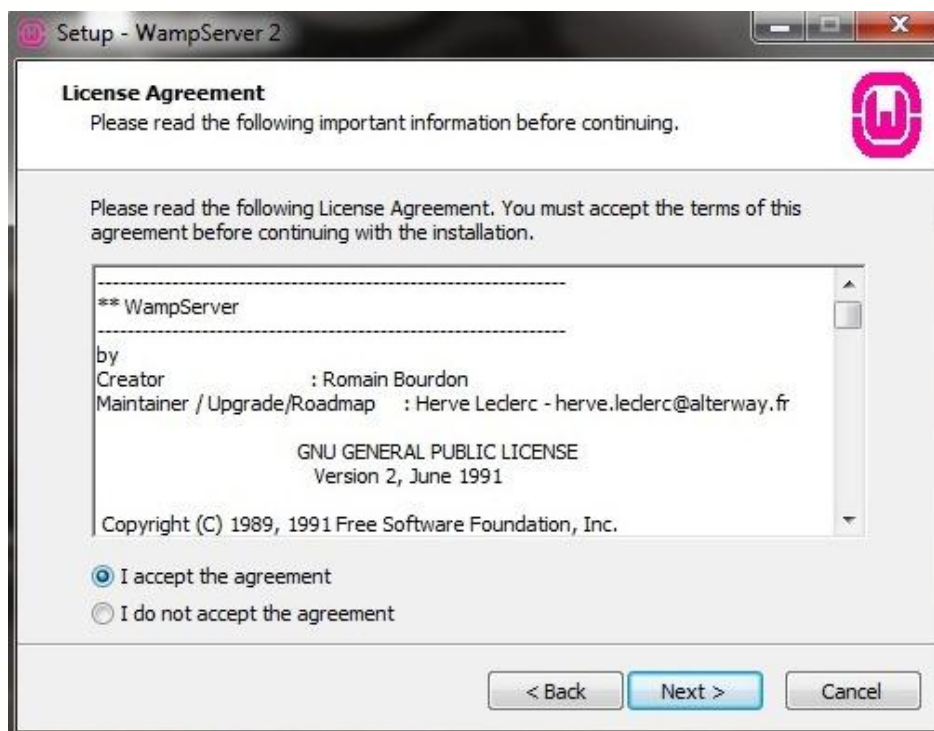
Baixe o WampServer no site: <http://www.wampserver.com/en/>.

Depois é só instalar seguindo os passos abaixo:

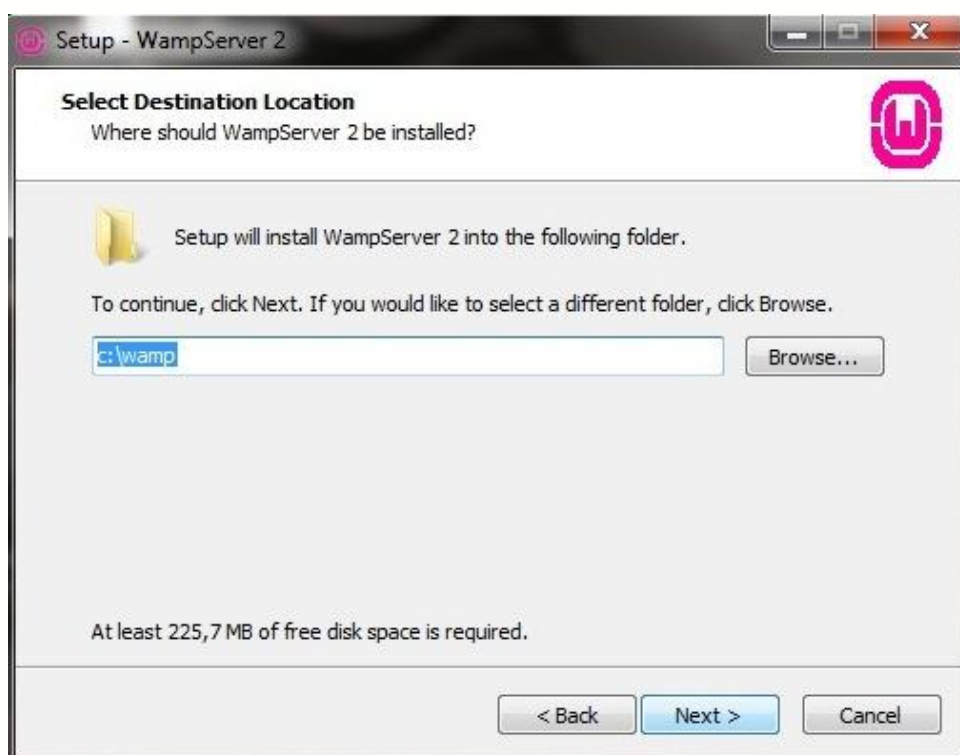
1. Clique em “NEXT >”.



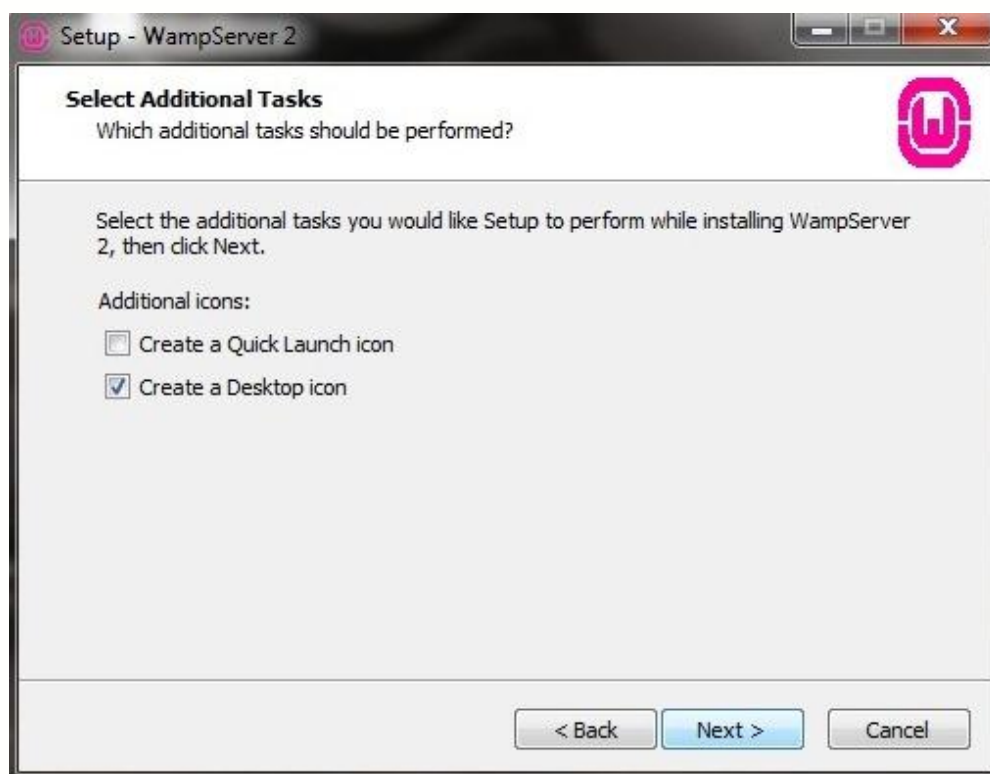
2. Marque "I accept the agreement" e clique em "NEXT >".



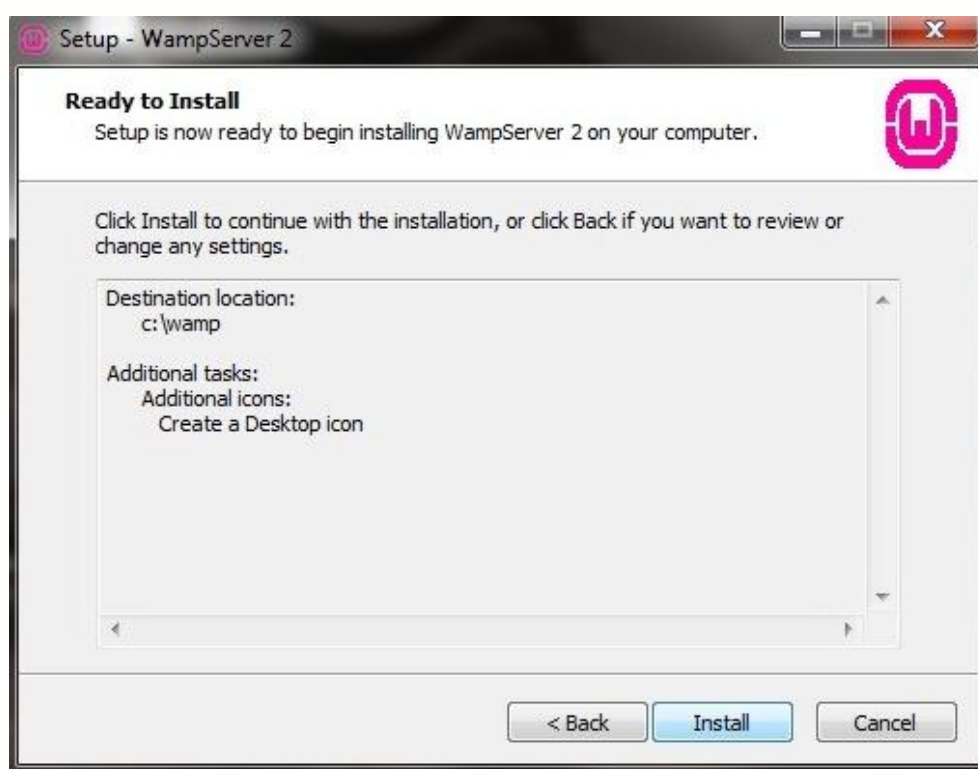
3. Escolha o local onde deseja instalar o WampServer. É recomendado deixar o caminho padrão "c:\wamp" e depois clicar em "NEXT >".



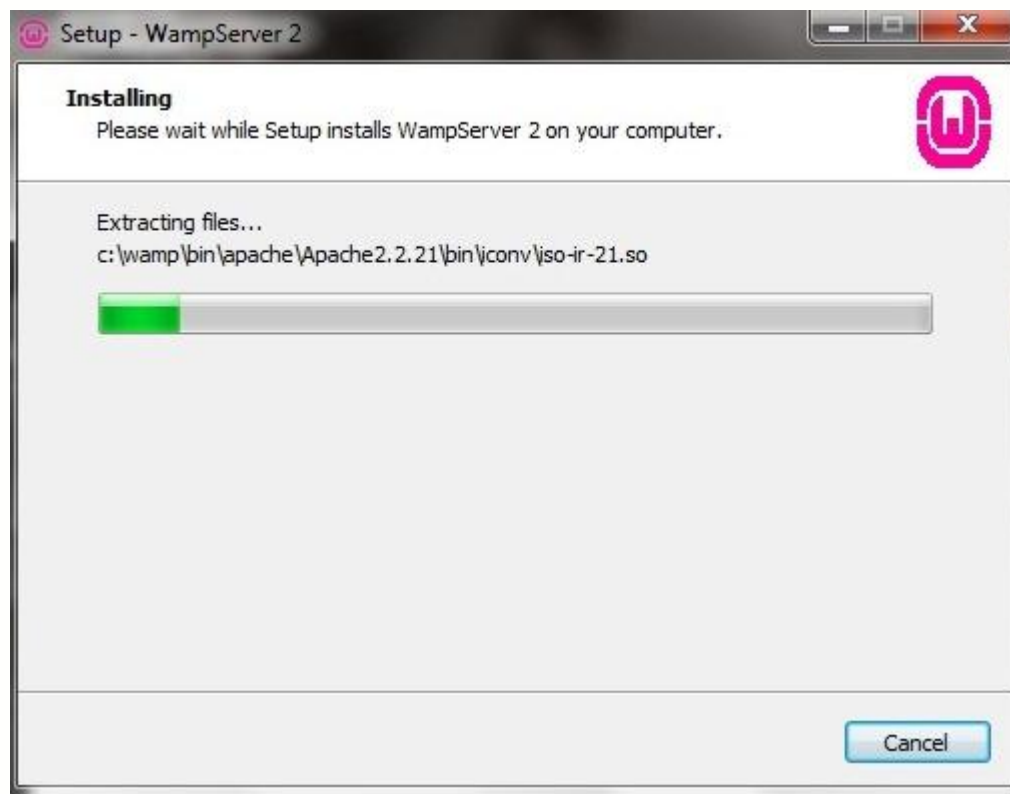
4. Selecione os ícones que deseja criar. É recomendado criar o ícone da área de trabalho e depois clicar em “NEXT >”.



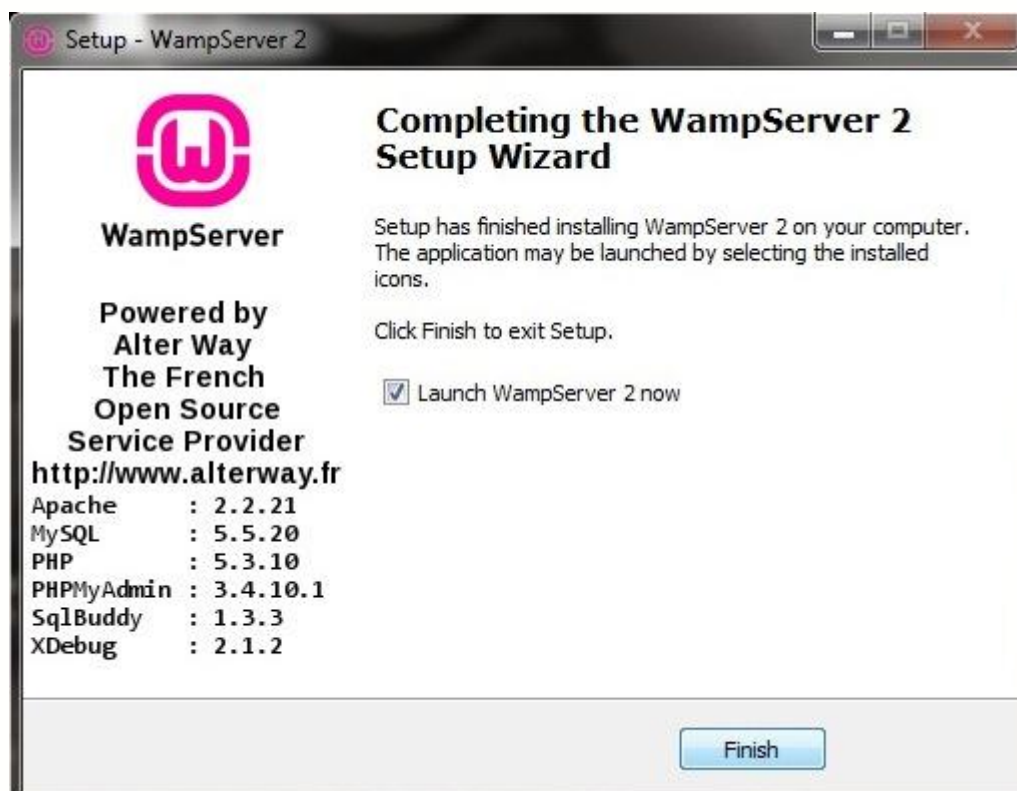
5. Clique em “INSTALL”.



6. Aguarde todo o processo de instalação terminar.



7. Depois que terminar a instalação clique em “FINISH”.



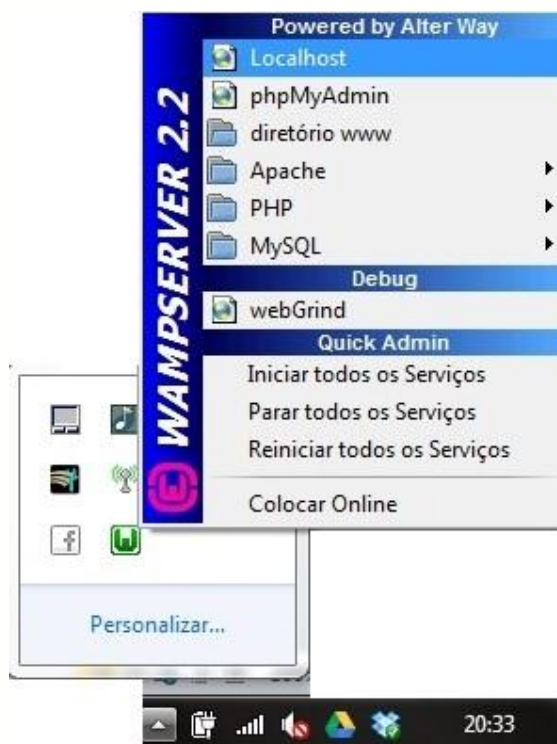
2.1.2 - CONFIGURANDO O WAMPSEVER

Para deixar o WampServer em português siga a imagem abaixo:

Clique com o botão direito sobre o ícone do WampServer, vá em “IDIOMA” e selecione “PORTUGUESE”.

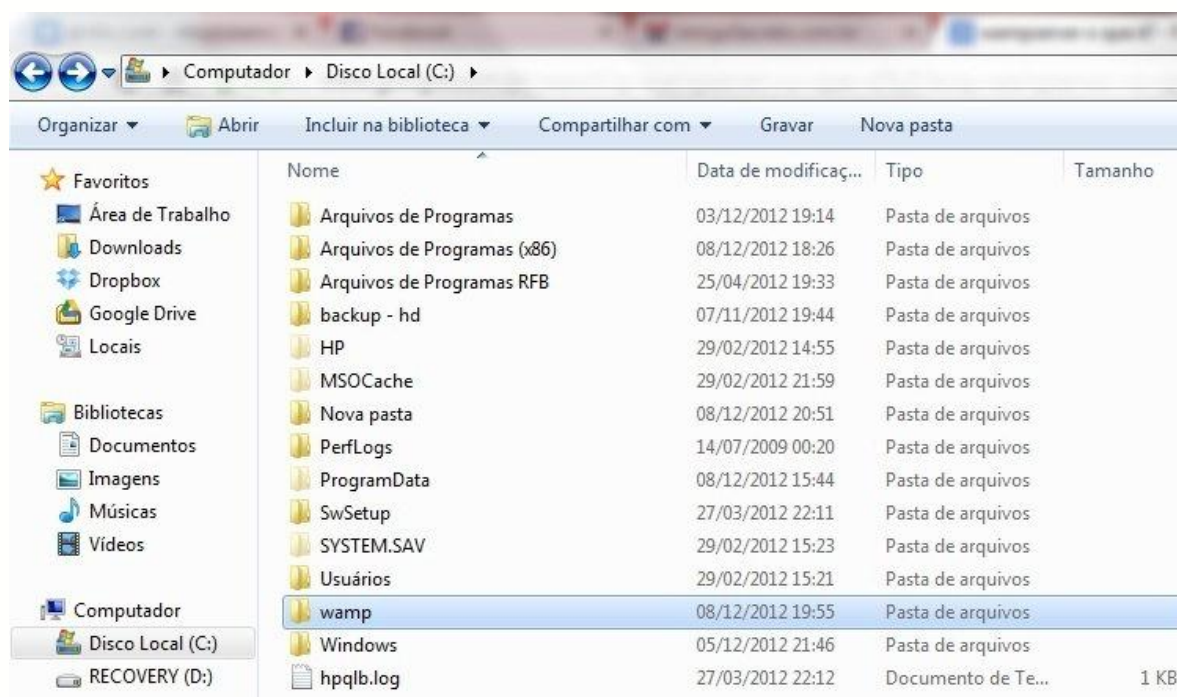


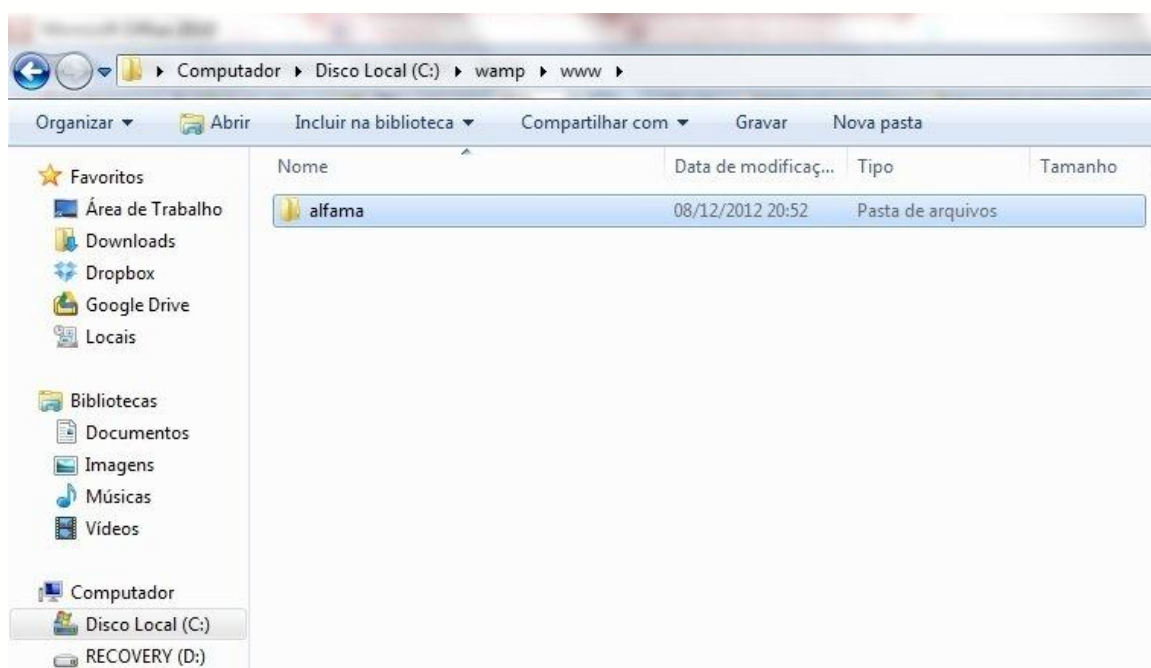
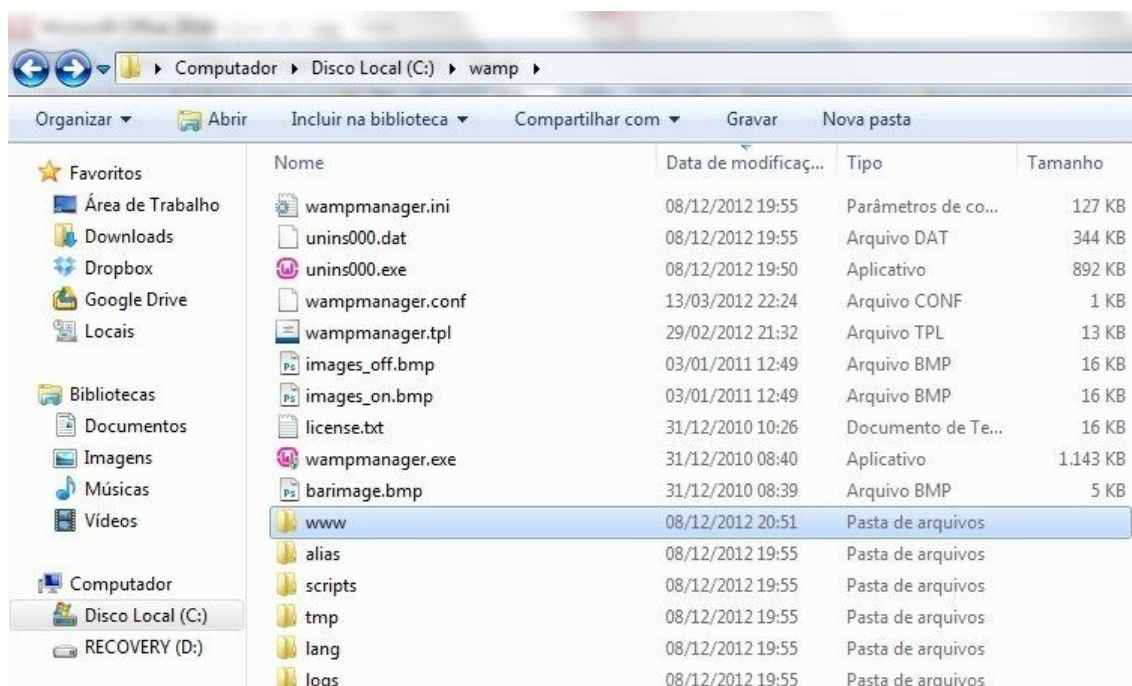
Clicando com o botão esquerdo sobre o ícone do WampServe, você verá os links para o Localhost, para o phpMyAdmin, para a pasta www, poderá abrir os arquivos de configuração do Apache, PHP e MySQL e poderá iniciar, parar e reiniciar os serviços do WampServer.

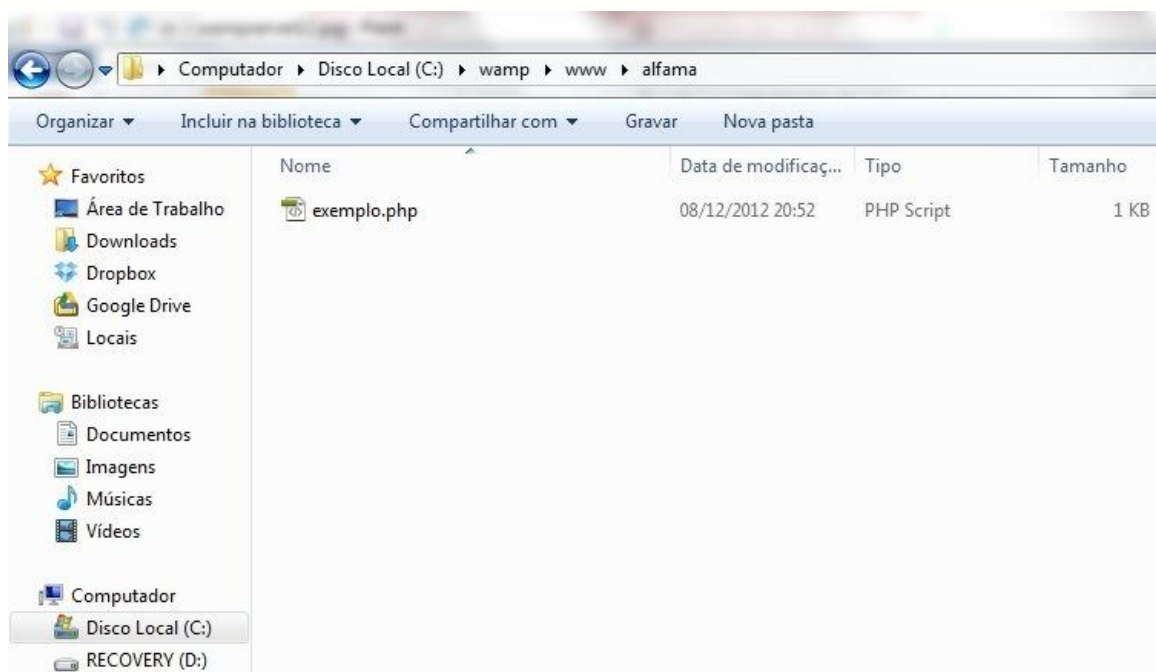


Crie uma pasta em C:\wamp\www e coloque todos os arquivos .php que você deseja executar dentro da pasta criada.

Exemplo:







Para executar seus arquivos .php, basta que o WampServer esteja sendo executado. Abra um navegador e coloque o endereço `http://localhost/nome_da_pasta/arquivo.php`.

2.2 - NOTEPAD++

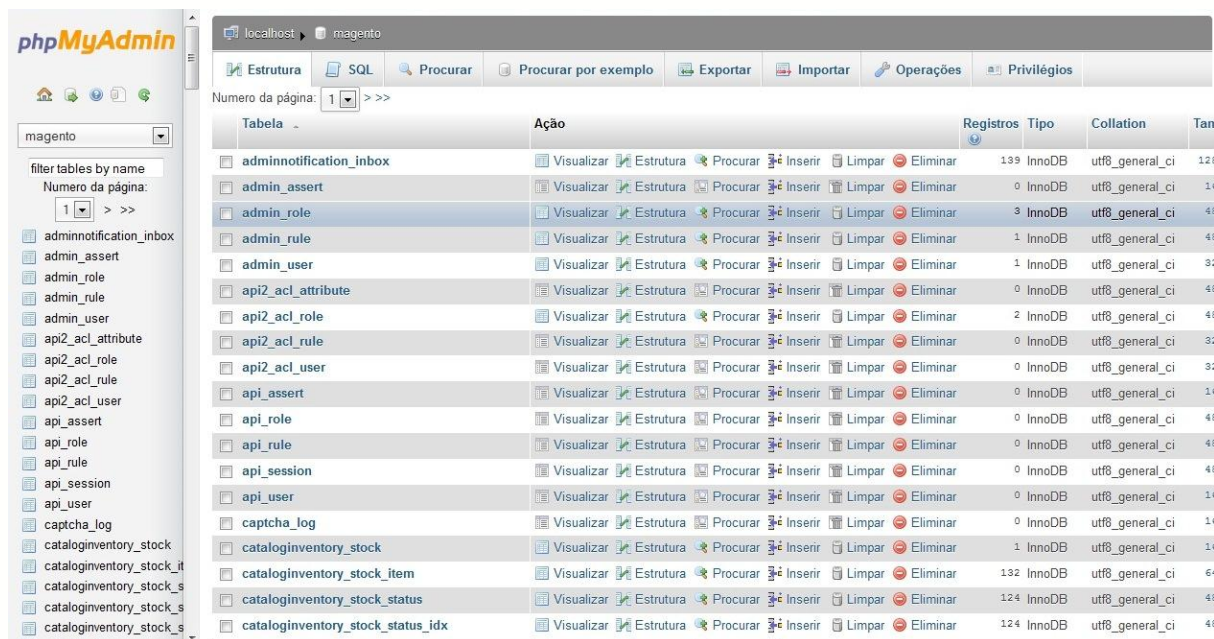
Notepad++ é um editor de texto e de código-fonte de código aberto sob a licença GPL. Suporta várias linguagens de programação rodando sob o sistema Microsoft Windows.

As linguagens de programação suportadas pelo Notepad++ são: C, C++, Java, C#, XML, HTML, PHP, JavaScript, makefile, ASCII art, doxygen, ASP, VB/VBScript, Unix Shell Script, BAT, SQL, Objective-C, CSS, Pascal, Perl, Python, Lua, Tcl, Assembly, Ruby, Lisp, Scheme, Smalltalk, PostScript e VHDL.

Ele suporta autocomplemento, busca e substituição de expressões regulares, divisão de tela, zoom, favoritos, etc. Tem suporte para macros e plugins.

2.3 - PHPMYADMIN

É um aplicativo web desenvolvido em PHP para administração do MySQL pela Internet. A partir deste sistema é possível criar e remover bases de dados, criar, remover e alterar tabelas, inserir, remover e editar campos, executar códigos SQL e manipular campos chaves.



2.4 – EXERCÍCIOS PROPOSTOS

1. O que é o WampServer?

2. O que é o PHPMyAdmin?

Capítulo 3

Introdução ao HTML

3 - INTRODUÇÃO AO HTML

3.1 - TAG'S HTML

Dentro de um documento HTML temos elementos de marcação denominados TAG'S . Uma TAG é definida através de seu nome cercado pelos sinais de “menor” (<) e “maior” (>) e normalmente tem uma TAG correspondente para finalização, com o mesmo nome e precedido por uma barra (/).

Ex.: <BODY> corpo do documento ... </BODY> (define o início e o fim do corpo do documento HTML).

Algumas TAG'S admitem de atributos (parâmetros) que alteram a maneira como o navegador deve interpretá-la e normalmente são colocados na TAG inicializadora.

Ex.: <BODY BACKGROUND=”fundo.gif”> corpo do documento </BODY> (define que o documento HTML terá como fundo a figura fundo.gif)

3.2 - TAG'S <HTML>

Todo documento HTML fica contido entre as tag's <HTML> e </HTML>. Dentro dessas tag's temos duas seções: HEAD e BODY que indicam o cabeçalho do documento e o corpo, respectivamente.

3.3 - TAG'S <HEAD>

Indicam o cabeçalho do documento e nelas estão contidas as informações de configuração da página.

3.4 - TAG'S <TITLE>

Essas tags devem ser utilizadas entre as tag's de cabeçalho <HEAD>. Determinam o título que irá ser apresentado na barra de título da janela do navegador.

3.5 - TAG'S <BODY>

Todo o conteúdo do documento HTML deve estar entre as tag's <BODY> e </BODY>.

3.6 - ESTRUTURA BÁSICA DO HTML

```
<HTML>

  <HEAD>

    <TITLE> TÍTULO DA PÁGINA</TITLE>

  </HEAD>

  <BODY>

    CORPO DA PÁGINA

  </BODY>

</HTML>
```

3.7 - FORMULÁRIO HTML

Um formulário é um modelo que permite ao usuário entrar com um conjunto de dados. O primeiro passo para fazer formulários é aprender as etiquetas que desenham as janelinhas de entrada de dados, para depois trabalharmos com os scripts, que são os programas que permitem a manipulação dos dados.

Um formulário é definido pela tag <form>, que irá conter uma sequência de elementos de entrada e de formatação do documento.

```
<form>

  Conteúdo do form

</form>
```

Os formulários podem conter qualquer formatação - parágrafos, listas, tabelas, imagens - exceto outros formulários. Em especial, colocamos dentro da marcação de <FORM> as formatações para campos de entrada de dados, que são três: <INPUT>, <SELECT> e <TEXTAREA>.

3.7.1 - <INPUT>

A tag mais usada em formulários é a tag <input>. O tipo de entrada é especificado com o atributo “type”. Todos os campos de entrada de dados têm um atributo NAME, ao qual associamos um nome, que será utilizado posteriormente pelo script. Alguns dos tipos mais comuns estão apresentados abaixo:

<form>

Login:

<input type="text" name="login">

Senha:

<input type="password" name="senha">

</form>

A maioria dos navegadores tem como largura padrão de um campo de texto 20 caracteres. Porém, com o atributo SIZE é possível definirmos o tamanho do campo. Também é possível delimitarmos o número de caracteres aceito no campo de dados com o atributo MAXLENGTH. Esses dois atributos apenas são válidos para campos TEXT e PASSWORD.

Ainda temos o atributo VALUE que pode ser usado para dar um valor inicial a um campo de tipo texto ou senha. Desse modo, se o usuário não preencher este campo, será adotado este valor padrão. Se o usuário quiser entrar com dados, ele simplesmente apaga o que já estiver escrito e escreve suas informações.

3.7.2 - RADIO

Botões Radio são utilizados quando se quer selecionar apenas uma opção dentre várias.

<form>

<input type="radio" name="sexo" value="masculino">Masculino


```
<input type=radio name="sexo" value="feminino" checked>Feminino
```

```
</form>
```

Note que apenas uma opção pode ser escolhida. Podemos perceber, também, que o atributo NAME tem o mesmo nome, porém, conforme a opção escolhida, seu VALUE é diferente.

3.7.3 - CHECKBOX

Utilizamos Checkboxes quando queremos selecionar mais de uma opção dentre várias.

```
<form>
```

```
<input type=checkbox name="check" value="propagandas" checked> Quero receber  
propagandas por email
```

```
<br>
```

```
<input type=checkbox name="check" value="atualizacoes" checked> Quero receber  
atualizações por email
```

```
</form>
```

3.7.4 - SELECT

A tag <SELECT> define uma seleção entre um conjunto de opções oferecidas ao usuário, estas opções são definidas na tag <OPTION>. Veja o exemplo abaixo:

```
<select>
```

```
<option value ="amarelo">Amarelo</option>
```

```
<option value ="azul">Azul</option>
```

```
<option value ="verde">Verde</option>
```

```
<option value ="vermelho">Vermelho</option>
```

```
</select>
```

Podemos definir o tamanho na tag <option> com o atributo SIZE ou também com o atributo SELECTED definir qual campo estará inicialmente selecionado. O padrão é que esteja selecionado o primeiro campo.

3.7.5 - TEXTAREA

A tag <TEXTAREA> define um campo texto, geralmente utilizado para comentários. Com os atributos COLS e ROWS definimos o tamanho do campo. Veja o exemplo abaixo:

```
<form>

    <textarea cols=40 rows=5 name="comentario">[Comentários]</textarea>

</form>
```

3.7.6 - BOTÃO

O atributo TYPE, que é igual a SUBMIT, apresenta o botão que causa o envio dos dados de entrada para o servidor, ou seja, quando o usuário clicar no botão "Enviar", todo o conteúdo do formulário é enviado para um outro arquivo.

```
<form>

    <input type=submit value="enviar">

</form>
```

3.8 - EXERCÍCIOS PROPOSTOS

1. O que são tag's HTML?

2. Qual é a estrutura básica do HTML?

Capítulo 4

Introdução ao PHP

4 - INTRODUÇÃO AO PHP

A linguagem PHP foi concebida durante o outono de 1994 por Rasmus Lerdorf. As primeiras versões não foram disponibilizadas, tendo sido utilizadas em sua homepage apenas para que ele pudesse ter informações sobre as visitas que estavam sendo feitas. A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como “*Personal Home Page Tools*” (ferramentas para página pessoal). Era composta por um sistema bastante simples que interpretava alguns macros e alguns utilitários que rodavam “por trás” das homepages: um livro de visitas, um contador e algumas outras coisas.

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de PHP/FI, o “FI” veio de um outro pacote escrito por Rasmus que interpretava dados de formulários HTML (*Form Interpreter*). Ele combinou os scripts do pacote *Personal Home Page Tools* com o FI e adicionou suporte a mSQL, nascendo assim o PHP/FI, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.

Estima-se que em 1996 o PHP/FI estava sendo usado por cerca de 15.000 sites pelo mundo, e em meados de 1997 esse número subiu para mais de 50.000. Nessa época houve uma mudança no desenvolvimento do PHP. Ele deixou de ser um projeto de Rasmus com contribuições de outras pessoas para ter uma equipe de desenvolvimento mais organizada. O interpretador foi reescrito por Zeev Suraski e Andi Gutmans, e esse novo interpretador foi a base para a versão 3.

4.1 - O QUE PODE SER FEITO COM O PHP?

Basicamente, qualquer coisa que pode ser feita por algum programa CGI pode ser feita também com PHP, como coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookies*.

PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como dBase, Interbase, mSQL, MySQL, Oracle, Sybase, PostgreSQL e vários outros. Construir uma página baseada em um banco de dados torna-se uma tarefa extremamente simples com PHP.

Além disso, PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir sockets e interagir com outros protocolos.

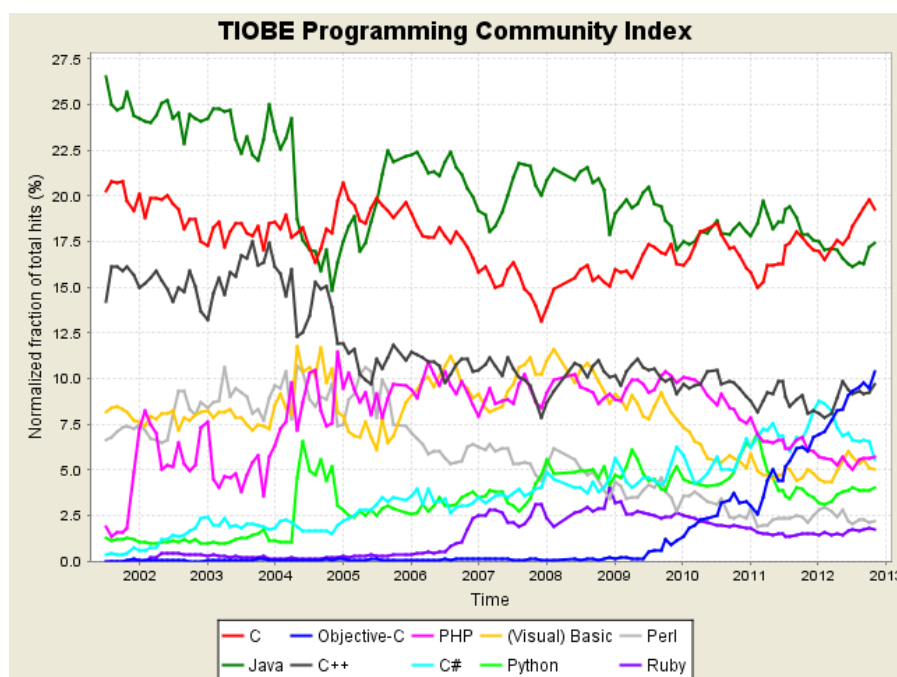
4.2 - CARACTERÍSTICAS DO PHP

Seu propósito principal é de implementar soluções web velozes, simples e eficientes.

Características:

- Velocidade e robustez.
- Orientação a objeto.
- Portabilidade - Independência de plataforma - *"write once run anywhere"* (escreva uma vez, rode em qualquer lugar).
- Tipagem fraca.
- Sintaxe similar à Linguagem C/C++ e o PERL.

Hoje, o PHP é a quinta linguagem de programação mais utilizada no mundo.



4.3 – EXERCÍCIOS PROPOSTOS

1. O que pode ser feito com o PHP?

2. Cite algumas características do PHP.

Capítulo 5

Sintaxe do PHP

5 - SINTAXE DO PHP

5.1 - DELIMITANDO O CÓDIGO PHP

O código PHP fica embutido no próprio HTML. O interpretador identifica quando um código é PHP pelas seguintes tag's:

```
<?php
```

```
    comandos
```

```
?>
```

```
<script language="php">
```

```
    comandos
```

```
</script>
```

```
<?
```

```
    comandos
```

```
?>
```

```
<%
```

```
    comandos
```

```
%>
```

O tipo de tag mais utilizado é o terceiro, que consiste em uma “abreviação” do primeiro. Para utilizá-lo, é necessário habilitar a opção short-tags na configuração do PHP. O último tipo serve para facilitar o uso por programadores acostumados à sintaxe de ASP. Para utilizá-lo, também é necessário habilitá-lo no PHP, através do arquivo de configuração php.ini.

5.2 - SEPARADOR DE INSTRUÇÕES

Entre cada instrução em PHP é preciso utilizar o ponto e vírgula, assim como em C, Perl e outras linguagens mais conhecidas. Na última instrução do bloco de script não é necessário o uso do ponto e vírgula, mas por questões estéticas recomenda-se o uso sempre.

5.3 - COMENTÁRIOS

Há dois tipos de comentários em código PHP:

- **Comentários de uma linha**

Marca como comentário até o final da linha ou até o final do bloco de código PHP – o que vier antes. Pode ser delimitado pelo caracter “#” ou por duas barras (//).

Exemplo:

```
<?php echo “teste”; #isto é um teste ?>
```

```
<?php echo “teste”; //este teste é similar ao anterior ?>
```

- **Comentários de mais de uma linha**

Tem como delimitadores os caracteres “/*” para o início do bloco e “*/” para o final do comentário. Se o delimitador de final de código PHP (?>) estiver dentro de um comentário, não será reconhecido pelo interpretador.

Exemplos:

```
<?php
```

```
/* Isto é um
```

```
comentário com mais
```

```
de uma linha. */
```

```
echo “teste”;
```

```
?>
```

5.4 - VARIÁVEIS

Toda variável em PHP tem seu nome composto pelo caracter \$ e uma string, que deve iniciar por uma letra ou o caracter “_”. PHP é *case sensitive*, ou seja, as variáveis \$vivas e \$VIVAS são diferentes. Por isso é preciso ter muito cuidado ao definir os nomes das variáveis. É bom evitar os nomes em maiúsculas, pois como veremos mais adiante, o PHP já possui algumas variáveis pré-definidas, cujos nomes são formados por letras maiúsculas.

Correto:

- \$nome = “João”;
- \$SobreNome = “Santos”;
- \$nome_curso = “PHP”;
- \$nome_curso2 = “PHP”;

Errado:

- \$8nome = “Erro!!”;

Não podem ser declaradas variáveis começando com número.

5.5 - EXERCÍCIOS PROPOSTOS

1. **Utilizando o Notepad++ crie um arquivo .php, empegando um dos delimitadores de código do PHP, e crie uma variável para armazenar seu nome.**

2. **Adicione comentários no arquivo criado na questão anterior.**

Capítulo 6

Comandos Básicos do PHP

6 - COMANDOS BÁSICOS DO PHP

6.1 - ECHO

Escreve na página HTML.

```
<?php  
  
    echo "Alfama Cursos";  
  
    echo "<br /><br />";  
  
    $curso = "PHP";  
  
    echo $curso;  
  
?>
```

6.2 - PRINT

Escreve na página HTML.

```
<?php  
  
    print "Alfama Cursos";  
  
    print "<br /><br />";  
  
    $curso = "PHP";  
  
    print $curso;  
  
?>
```

Obs.: O ECHO é mais usado do que o PRINT.

Podem ser utilizadas tag's HTML na string e imprimir essa string na tela usando o ECHO ou o PRINT.

```
<?php  
  
    echo "<h1>Alfama Cursos</h1>";
```

```
$curso = "<h2>PHP</h2>";
```

```
echo $curso;
```

```
?>
```

Utilizando a short-tag php juntamente com um “=” é outra maneira de escrever na página HTML.

```
<?= “Alfama Cursos”; ?>
```

```
<?php
```

```
$curso = “PHP”;
```

```
?>
```

```
<?= $nome; ?>
```

6.3 - CONCATENAÇÃO

Para concatenar duas strings ou duas variáveis é utilizado o “.”.

```
<?php
```

```
echo “Alfama”.”Cursos”;
```

```
echo “<br /><br />”;
```

```
$nome = “João”;
```

```
$sobreNome = “Santos”;
```

```
echo $nome.$sobreNome;
```

```
?>
```

Se a string estiver entre “ ” (aspas duplas) e uma variável for colocada dentro dessa string, a variável será concatenada a string, se estiver entre ‘ ’ (aspas simples), a string receberá apenas o nome da variável.

```
<?php
```

```
$sobreNome = "Santos";

echo "João $sobreNome"; //Retorna: João Santos

echo "<br /><br />";

echo 'João $sobreNome'; //Retorna: João $sobreNome

?>
```

6.4 – EXERCÍCIOS PROPOSTOS

1. Utilizando o Notepad++, crie um arquivo .php, crie uma variável para armazenar seu nome e uma outra para armazenar seu sobrenome. Escreva na tela essas variáveis.

2. Utilizando o arquivo .php criado na questão anterior, concatene as duas variáveis (nome e sobrenome) e escreva na tela.

Capítulo 7

Tipos de Dados

7 - TIPOS DE DADOS

O PHP tem tipagem fraca, basta atribuir um valor à variável e o PHP automaticamente, de acordo com o tipo do dado associado, define o tipo da variável.

O PHP suporta os seguintes tipos:

- Números inteiros.
- Números reais.
- Caracteres alfanuméricos.
- Lógicos.
- Arrays.
- Objetos.

7.1 - INTEIROS (INTEGER E LONG)

São os números inteiros, negativos e positivos.

```
<?php
```

```
$numero = 18;
```

```
echo $numero;
```

```
echo "<br /><br />";
```

```
$numero = -18;
```

```
echo $numero;
```

```
?>
```

7.2 - REAIS (FLOAT E DOUBLE)

São os números com ponto flutuante, negativos e positivos.

```
<?php
```

```
$numero = 2.18;
```

```

echo $numero;

echo "<br /><br />";

$numero = -2.18;

echo $numero;

?>

```

7.3 - CARACTERES ALFANUMÉRICOS (STRING)

São os caracteres alfanuméricos.

```

<?php

$curso = "Linguagem de programação PHP - 2012";

echo $ curso;

?>

```

7.4 - LÓGICOS (BOOLEANO)

São os tipos lógicos, verdadeiro (true) ou falso (false).

O valor falso é representado por 0 e o valor verdadeiro é representado por 1.

```

<?php

$verdadeiro = true;

$falso = false;

echo "Verdadeiro: ".$verdadeiro."<br />Falso: ".$falso

?>

```

7.5 - ARRAYS

Grupos de elementos separados por índices.

```
<?php

$array["nome"] = "Alfama";

$array["curso"] = "PHP";

$array[0] = 10;

$array[1] = 20;

print_r($array);

?>
```

7.6 - OBJETOS

Grupos de atributos e métodos.

```
<?php

class curso {

    function getNome() {

        echo "PHP";

    }

}

$objeto = new curso;

$objeto->curso();

?>
```

7.7 - EXERCÍCIO PROPOSTO

1. Quais são os tipos de dados suportados pelo PHP?

Capítulo 8

Array

8 - ARRAY

Arrays em PHP podem ser observados como mapeamentos ou como vetores indexados. Mais precisamente, um valor do tipo Array é um dicionário onde os índices são as chaves de acesso. Vale ressaltar que os índices podem ser valores de qualquer tipo e não somente inteiros. Inclusive, se os índices forem todos inteiros, estes não precisam formar um intervalo contínuo.

Como a checagem de tipos em PHP é dinâmica, valores de tipos diferentes podem ser usados como índices de Array, assim como os valores mapeados também podem ser de diversos tipos.

```
<?php
```

```
    $cor[4] = "vermelho";
```

```
    $cor[2] = "verde";
```

```
    $cor["teste"] = 1;
```

```
?>
```

```
<?php
```

```
    $cor = array(1 => "Vermelho, 2 => "Verde, 3 => "Azul", "teste" => 1);
```

```
    $carro = array("Gol", "Fiesta", "Uno");
```

```
    $nome[] = "João";
```

```
    $nome[] = "Maria";
```

```
    $nome[] = "José";
```

```
?>
```

8.1 - PRINT_R

Imprime todos os índices do Array.

```
<?php

$cor = array(1 => "Vermelho, 2 => "Verde, 3 => "Azul", "teste" => 1);

print_r($cor);

?>
```

8.2 - COUNT

Retorna o total de elementos do Array.

```
<?php

$cor = array(1 => "Vermelho, 2 => "Verde, 3 => "Azul", "teste" => 1);

echo "Total de elementos: ".count($cor);

?>
```

8.3 – EXERCÍCIOS PROPOSTOS

1. O que é Array?

2. Crie um Array para armazenar os seguintes valores:

- a) Gol
- b) Palio
- c) Uno
- d) Fiesta

Capítulo 9

Operadores

9 - OPERADORES

9.1 - ARITMÉTICOS

Só podem ser utilizados quando os operandos são números (integer ou float). Se forem de outro tipo, terão seus valores convertidos antes da realização da operação.

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto

```
<?php
```

```
    $nota1 = 10;
```

```
    $nota2 = 8;
```

```
    $media = ($nota1 + $nota2) / 2;
```

```
    echo "Média: ".$media;
```

```
?>
```

9.2 - ATRIBUIÇÃO

Existe um operador básico de atribuição e diversos derivados. Sempre retornam o valor atribuído. No caso dos operadores derivados de atribuição, a operação é feita entre os dois operandos, sendo atribuído o resultado para o primeiro. A atribuição é sempre por valor, e não por referência.

=	Atribuição simples
+=	Atribuição com adição
-=	Atribuição com subtração
*=	Atribuição com multiplicação
/=	Atribuição com divisão
%=	Atribuição com resto
.=	Atribuição com concatenação

```
<?php
```

```
$media = 10;
```

```
$media += 8;
```

```
$media /= 2;
```

```
echo "Média: ".$media;
```

```
?>
```

9.3 - COMPARAÇÃO

As comparações são feitas entre dois valores. Sempre retornam um valor booleano (verdadeiro ou falso).

==	Igual
!=	Diferente
===	Exatamente igual
!==	Exatamente diferente
<	Menor que
>	Maior que
<=	Menor ou igual
=>	Maior ou igual

Obs.: Os comparadores igual (==) e diferente (!=) comparam apenas o valor do dado, já os comparadores exatamente igual (===) e exatamente diferente (!==), além de comparar o valor dos dados compara também o tipo dos dados.

```
<?php

$nota1 = 10;

$nota2 = 8;

if($nota >= $nota2) {

    echo $nota1." é maior ou igual a ".$nota2;

}

?>
```

9.4 - LÓGICOS

Utilizados para fazer comparações lógicas. Retornam sempre os valores verdadeiro ou falso.

&&	E
and	E
 	OU
or	OU
xor	OU exclusivo
!	Negação

```
<?php

$nota1 = 10;

$nota2 = 8;

$media = 6;

if($nota1 > $media && $nota2 > $media) {

    echo "O aluno tirou notas acima da média";

}
```

```
}
```

```
?>
```

9.5 - INCREMENTO E DECREMENTO

Podem ser utilizados de duas formas: antes ou depois da variável. Quando utilizado antes, retorna o valor da variável antes de incrementá-la ou decrementá-la. Quando utilizado depois, retorna o valor da variável já incrementado ou decrementado.

++	Incremento
--	Decremento

```
<?php
```

```
$numero = 1;
```

```
$numero++;
```

```
echo $numero;
```

```
?>
```

9.6 – EXERCÍCIO PROPOSTO

1. Crie duas variáveis, \$x = 10 e \$y = 8. Utilizando os operadores, armazene o valor das operações entre as duas variáveis criadas na variável \$valor e escreva o valor de \$valor na tela.

Exemplo:

```
$x = 10;
```

```
$y = 8;
```

```
$valor = $x + $y;
```

```
echo $valor;
```

Capítulo 10

Estruturas de Controle (Parte I)

10 - ESTRUTURAS DE CONTROLE (PARTE I)

As estruturas que veremos a seguir são comuns para as linguagens de programação imperativas, bastando, portanto, descrever a sintaxe de cada uma delas, resumindo o funcionamento.

Também chamados de condicionais, os comandos de seleção permitem executar comandos ou blocos de comandos com base em testes feitos durante a execução.

10.1 - BLOCOS

Um bloco consiste de vários comandos agrupados com o objetivo de relacioná-los com determinado comando ou função.

Blocos em PHP são delimitados pelos caracteres “{” e “}”.

```
<?php
```

```
    if ($x == $y){
```

```
        Códigos PHP
```

```
    }
```

```
?>
```

10.2 - IF

O mais trivial dos comandos condicionais é o if. Ele testa a condição e executa o comando indicado se o resultado for true (valor diferente de zero).

Sintaxe:

```
if (expressão) {
```

```
    comandos;
```

```
}
```

Exemplo:

```
<?php

    $x = 10;

    $y = 10;

    if ($x == $y){

        echo "O número $x é igual ao número $y";

    }

?>
```

10.3 - ELSE

O else é um complemento opcional para o if. Se utilizado, o comando será executado se a expressão retornar o valor false (zero).

Sintaxe:

```
if (expressão) {

    comandos;

} else {

    comandos;

}
```

Exemplo:

```
<?php

    $x = 10;

    $y = 20;

    if ($x == $y){

        echo "O número $x é igual ao número $y";

    }
```

```

    } else {

        echo "O número $x é diferente ao número $y";

    }

?>

```

10.4 - ELSEIF

Em determinadas situações é necessário fazer mais de um teste e executar condicionalmente diversos comandos ou blocos de comandos. Para facilitar foi criado o comando, também opcional elseif.

Ele tem a mesma função de um else e um if usados sequencialmente.

Sintaxe:

```

if (expressão) {

    comandos;

} elseif(expressão) {

    comandos;

} else {

    comandos;

}

```

Exemplo:

```

<?php

    $x = 10;

    $y = 10;

    if ($x > $y){

        echo "O número $x é maior que o número $y";
    }

```

```

    } elseif($x < $y) {

        echo "O número $x é menor que o número $y";

    } else {

        echo "O número $x é igual ao número $y";

    }

?>

```

10.5 - SWITCH

O comando switch atua de maneira semelhante a uma série de comandos if na mesma expressão.

Frequentemente, o programador pode querer comparar uma variável com diversos valores, e executar um código diferente a depender de qual valor é igual ao da variável. Quando isso for necessário, deve-se usar o comando switch. O exemplo seguinte mostra dois trechos de código que fazem a mesma coisa, sendo que o primeiro utiliza uma série de if's e o segundo utiliza switch.

Sintaxe:

```

switch (variavel) {

    case "comparação 1":

        comandos;

        break;

    case "comparação 2":

        comandos;

        break;

    case "comparação 2":

        comandos;

```

```

        break;

    }

```

É importante compreender o funcionamento do switch para não cometer enganos. O comando switch testa linha a linha os cases encontrados, e a partir do momento que encontra um valor igual ao da variável testada, passa a executar todos os comandos seguintes, mesmo os que fazem parte de outro teste, até o fim do bloco. Por isso usa-se o comando break, quebrando o fluxo e fazendo com que o código seja executado da maneira desejada.

Exemplo:

```

<?php

    switch ($i) {

        case 0:

            echo "i é igual a zero";

        case 1:

            echo "i é igual a um";

        case 2:

            echo "i é igual a dois";

    }

?>

```

No exemplo acima, se \$i for igual a zero, os três comandos “echo” serão executados. Se \$i for igual a 1, os dois últimos “echo” serão executados. O comando só funcionará da maneira desejada se \$i for igual a 2. Para todo o switch ser executado de maneira correta, basta colocar o break no final de cada case.

```

<?php

    switch ($i) {

        case 0:

```

```

        echo "i é igual a zero";

        break;

    case 1:

        echo "i é igual a um";

        break;

    case 2:

        echo "i é igual a dois";

        break;

    }

?>

```

Obs.: Em outras linguagens que implementam o comando switch, ou similar, os valores a serem testados só podem ser do tipo inteiro. Em PHP é permitido usar valores do tipo string como elementos de teste do comando switch.

10.6 - EXERCÍCIOS PROPOSTOS

1. O que são Estruturas de Controle?

2. Crie duas variáveis, \$x e \$y, atribua um valor numérico a essas variáveis. Utilizando a estrutura de controle IF, ELSEIF e ELSE escreva na tela qual numero é maior, qual é menor ou se são iguais.

Capítulo 11

Estruturas de Controle (Parte II)

11 - ESTRUTURAS DE CONTROLE (PARTE II)

11.1 - WHILE

O while é o comando de repetição (laço) mais simples. Ele testa uma condição e executa um comando, ou um bloco de comandos, até que a condição testada seja falsa.

A expressão só é testada a cada vez que o bloco de instruções termina, além do teste inicial. Se o valor da expressão passar a ser false no meio do bloco de instruções, a execução segue até o final do bloco. Se no teste inicial a condição for avaliada como false, o bloco de comandos não será executado.

Sintaxe:

```
while (expressão) {  
    comandos;  
}
```

Exemplo:

```
<?php  
  
$x = 0;  
  
while ($x <= 9){  
    echo $x++;  
}  
  
?>
```

11.2 - DO...WHILE

O laço do...while funciona de maneira bastante semelhante ao while, com a simples diferença que a expressão é testada ao final do bloco de comandos.

Sintaxe:

```
do {  
    comandos;  
} while (expressão)
```

Exemplo:

```
<?php  
    $x = 0;  
    do {  
        echo $x++;  
    } while ($x <= 9)  
?>
```

11.3 - FOR

O tipo de laço mais complexo é o for.

Sintaxe:

```
for (inicialização;condição;incremento) {  
    comandos;  
}
```

As três expressões que ficam entre parênteses têm as seguintes finalidades:

- **Inicialização:** comando ou sequência de comandos a serem realizados antes do início do laço. Serve para inicializar variáveis.
- **Condição:** expressão booleana que define se os comandos que estão dentro do laço serão executados ou não. Enquanto a expressão for verdadeira (valor diferente de zero) os comandos serão executados.

- **Incremento:** comando executado ao final de cada execução do laço.

Exemplo:

```
<?php
    for ($x = 0; $x <= 9;$x++) {
        echo $x;
    }
?>
```

11.4 - BREAK

O comando break pode ser utilizado em laços de **do**, **for** e **while**, além do uso já visto no comando switch. Ao encontrar um break dentro de um desses laços, o interpretador PHP para imediatamente a execução do laço, seguindo normalmente o fluxo do script.

Exemplo:

```
<?php
    switch ($i) {
        case 0:
            echo "i é igual a zero";
            break;
        case 1:
            echo "i é igual a um";
            break;
        case 2:
            echo "i é igual a dois";
            break;
```

```

    }

?>

```

```

<?php
    for ($i < 100;) {

        if ($i == 20) { break; }

        $i++;

        echo $i ;

    }

?>

```

11.5 - CONTINUE

O comando continue também deve ser utilizado no interior de laços, e funciona de maneira semelhante ao break, com a diferença que o fluxo ao invés de sair do laço volta para o início dele.

Exemplo:

```

<?php
    for ($i = 0; $i < 100; $i++) {

        if ($i % 2 > 0) {

            continue;

        }

        echo $i;

    }

?>

```

11.6 - EXERCÍCIOS PROPOSTOS

1. Explique os comandos **BREAK** e **CONTINUE**.

2. Escreva os números de 0 até 100, utilizando o **WHILE**, **DO...WHILE** e **FOR**.

Capítulo 12

Estruturas de Controle (Parte III)

12 - ESTRUTURAS DE CONTROLE (PARTE III)

12.1 - EXEMPLO DE UTILIZAÇÃO DO IF

Verifica o valor da variável e executa o bloco de código quando o valor da expressão é verdadeiro.

```
<?php

$cor = "branco";

if ($cor == "vermelho") {

    echo "A variável contém o valor 'vermelho'.";

} else if ($cor == "azul") {

    echo "A variável contém o valor 'azul'.";

} else if ($cor == "amarelo") {

    echo "A variável contém o valor 'amarelo'.";

} else {

    echo "O valor da variável não foi identificado.";

}

?>
```

Verifica se o número é par ou ímpar e imprime o resultado.

```
<?php

$numero = 10;

if (($numero % 2) == 0) {

    echo $numero. " é um número par.";

} else {
```

```

        echo $numero.“ é um número impar.”;

    }

?>

```

12.2 - EXEMPLO DE UTILIZAÇÃO DO SWITCH

Imprime o mês por extenso.

```

<?php

$mes = 8;

switch ($mes) {

    case 1:

        echo “Janeiro”;

        break;

    case 2:

        echo “Fevereiro”;

        break;

    case 3:

        echo “Março”;

        break;

    case 4:

        echo “Abril”;

        break;

    case 5:

        echo “Maio”;

```

```
break;

case 6:

    echo "Junho";

break;

case 7:

    echo "Julho";

break;

case 8:

    echo "Agosto";

break;

case 9:

    echo "Setembro";

break;

case 10:

    echo "Outubro";

break;

case 11:

    echo "Novembro";

break;

case 12:

    echo "Dezembro";

break;
```

```
}
```

?>

12.3 - EXEMPLO DE UTILIZAÇÃO DO WHILE, DO...WHILE E FOR

Retorna todos os números pares de 0 até 100.

```
<?php
```

```
$numero = 0;
```

```
while ($numero <= 100) {
```

```
    if(($numero % 2) == 0) {
```

```
        echo $numero;
```

```
        echo "<br />";
```

```
    }
```

```
    $numero++;
```

```
}
```

```
?>
```

Esse mesmo exemplo também pode ser feito utilizando o do...while.

```
<?php
```

```
$numero = 0;
```

```
do {
```

```
    if(($numero % 2) == 0) {
```

```
        echo $numero;
```

```
        echo "<br />";
```

```
    }
```

```
    $numero++;
```

```
} while ($numero <= 100);
```

```
?>
```

Uma terceira maneira de imprimir os números pares de 0 até 100 é utilizando o for.

```
<?php
```

```
for($numero = 0; $numero <= 100; $numero++) {
```

```
    if(($numero % 2) == 0) {
```

```
        echo $numero;
```

```
        echo "<br />";
```

```
    }
```

```
}
```

```
?>
```

12.4 - EXEMPLO DE UTILIZAÇÃO DO FOR

Percorrer e imprimir todos os elementos de um Array utilizando o for.

```
<?php
```

```
$nomes = array("João", "Maria", "José");
```

```
for($i = 0; $i < count($nomes); $i++) {
```

```
    echo $nomes[$i];
```

```
    echo "<br />";
```

```
}
```

```
?>
```

12.5 - EXERCÍCIOS PROPOSTOS

1. Crie um Array, adicione os valores a esse Array e utilizando o FOR, percorra e escreva na tela todos os valores do Array criado.

Capítulo 13

Funções

13 - FUNÇÕES

13.1 - DEFINIÇÃO DE FUNÇÕES

A sintaxe básica para definir uma função é:

```
function nome_da_função([arg1, arg2, arg3]) {  
  
    Comandos;  
  
    [return <valor de retorno>];  
  
}
```

Qualquer código PHP válido pode estar contido no interior de uma função. Como a checagem de tipos em PHP é dinâmica, o tipo de retorno não deve ser declarado, sendo necessário que o programador esteja atento para que a função retorne o tipo desejado. É recomendável que esteja tudo bem documentado para facilitar a leitura e compreensão do código.

13.2 - RETORNO

Toda função pode opcionalmente retornar um valor, ou simplesmente executar os comandos e não retornar valor algum.

Não é possível que uma função retorne mais de um valor, mas é permitido fazer com que uma função retorne um valor composto, como Arrays.

13.3 - PARÂMETROS

É possível passar parâmetros para uma função. Eles devem ser declarados logo após o nome da função, entre parênteses, e tornam-se variáveis pertencentes ao escopo local da função. A declaração do tipo de cada argumento também é utilizada apenas para efeito de documentação.

Exemplo:

```
<?php
```

```
function imprime($texto){
    echo $texto;
}

imprime("teste de funções");

?>
```

13.4 - PARÂMETROS COM VALORES PRÉ-DEFINIDOS (DEFAULT)

Em PHP é possível ter valores default para parâmetros de funções, ou seja, valores que serão assumidos em caso de nada ser passado no lugar do parâmetro. Quando algum parâmetro é declarado desta maneira, a passagem do mesmo na chamada da função torna-se opcional.

Exemplo:

```
<?php

function teste($curso = "JavaScript") {
    echo $curso;
}

teste(); //imprime "JavaScript"

teste("PHP"); //imprime "PHP"

?>
```

Quando a função tem mais de um parâmetro, o que tem valor default deve ser declarado por último:

```
<?php

function teste($figura = "circulo", $cor) {
    echo "a figura é um ".$figura." de cor ".$cor;
}

teste("azul");
```

?>

A função não vai funcionar da maneira esperada, ou seja, imprime o texto: **A figura é um azul de cor.**

A declaração correta é:

<?php

```
function teste($cor, $figura = "circulo") {
    echo "a figura é um ".$figura." de cor ".$cor;
}
teste("azul");
```

?>

Aqui a função funciona da maneira esperada, ou seja, imprime o texto: **A figura é um círculo de cor azul.**

Capítulo 14

Classes e Objetos

14 - CLASSES E OBJETOS

14.1 - CLASSES

Uma classe é um conjunto de variáveis e funções relacionadas a essas variáveis. Uma vantagem da utilização é poder usufruir do recurso de encapsulamento de informação. Com o encapsulamento, o usuário de uma classe não precisa saber como ela é implementada, bastando para a utilização conhecer a interface, ou seja, as funções disponíveis. Uma classe é um tipo, portanto não pode ser atribuída a uma variável.

Sintaxe:

```
class Nome_da_classe {

    var $variavel1;

    var $variavel2;

    function funcao1 ($parametro) {

        /* === corpo da função === */

    }

}
```

Exemplo:

```
<?php

class conta {

    var $saldo;

    function saldo() {

        return $this->saldo;

    }

}
```

?>

14.2 - OBJETOS

Como foi dito anteriormente, classes são tipos, e não podem ser atribuídas às variáveis. Variáveis do tipo de uma classe são chamadas de objetos, e devem ser criadas utilizando o operador new, seguindo o exemplo abaixo:

```
$variavel = new $nome_da_classe;
```

Para utilizar as funções definidas na classe, deve ser utilizado o operador “->”, como no exemplo:

```
$variavel->funcao1($parametro);
```

14.3 - SUBCLASSES

Uma classe pode ser uma extensão de outra. Isso significa que ela herdará todas as variáveis e funções da outra classe, e ainda terá as funções e variáveis que forem adicionadas na subclasse. Em PHP, não é permitido utilizar herança múltipla, ou seja, uma classe pode ser extensão de apenas uma outra. Para criar uma classe estendida, ou derivada de outra, deve ser utilizada a palavra reservada extends.

Exemplo:

```
<?php

class novaconta extends conta {

    var $numero;

    function numero() {

        return $this->numero;

    }

}
```

?>

A classe acima é derivada da classe conta, tendo as mesmas funções e variáveis, com a adição da variável \$numero e a função numero().

14.4 - CONSTRUTORES

Um construtor é uma função definida na classe que é automaticamente chamada no momento em que a classe é instanciada (através do operador new). O construtor deve ter o mesmo nome que a classe a que pertence.

Exemplo:

```
<?php

class conta {

    var $saldo;

    function conta () {

        $this->saldo = 0;

    }

    function saldo() {

        return $this->saldo;

    }

}

?>
```

Capítulo 15

Get, Post e File

15 - GET, POST E FILE

Utilizando os formulários HTML, o PHP pode recuperar os dados através da variável pré-definida `$GET` e `$_POST`, bastando indicar o nome do campo do formulário. No comando action do formulário HTML, deve-se indicar a página PHP que irá receber os valores. O mesmo documento pode conter o código e o formulário.

O PHP permite que sejam enviados arquivos para o servidor utilizando um formulário HTML e a variável pré-definida `$_FILE`.

15.1 - GET

Com o método GET, os dados podem ser visualizados na URL do browser. Para recuperar estes dados, deve-se usar a variável pré-definida `$_GET`.

Exemplo:

```
<html>

    <head>

        <title>Alfama Cursos</title>

    </head>

    <body>

        <?php

            if($_GET["nome"] && $_GET["sobre_nome"]) {

                echo "Olá ".$_GET["nome"]." ".$_GET["sobre_nome"];

            } else {

                echo "Digite o seu nome.<p>";

            }

        ?>

        <form method="get" action="teste.php">
```

Primeiro Nome: <input type="text" name="nome">

Sobrenome: <input type="text" name="sobre_nome">

<input type="submit" value="Enviar">

</form>

</body>

</html>

15.2 - POST

Com o método POST, os dados não podem ser visualizados na URL do browser. Para recuperar estes dados, deve-se usar a variável pré-definida \$_POST.

Exemplo:

<html>

<head>

<title>Alfama Cursos</title>

</head>

<body>

<?php

if (\$_POST["nome"] && \$_POST["sobre_nome"]) {

echo "Olá " . \$_POST["nome"] . " " . \$_POST["sobre_nome"];

} else {

```

        echo "Digite o seu nome.<p>";

    }

?>

<form method="post" action="teste.php">

    Primeiro Nome: <input type="text" name="nome">

    <br>

    Sobrenome: <input type="text" name="sobre_nome">

    <br>

    <br>

    <input type="submit" value="Enviar">

</form>

</body>

</html>

```

15.3 - FILES

Todas as informações sobre o arquivo enviado ficam armazenadas na variável \$_FILES. O comando que trata o envio do arquivo é move_uploaded_file.

Exemplo:

```

<html>

    <head>

        <title>Alfama Cursos</title>

    </head>

    <body>

```

```

<?php

if ($_FILES["arquivo"]["name"]){

    $uploadaddir = "c:\\temp\\";

    $arquivo = $uploadaddir.$_FILES["arquivo"]["name"];

    if (move_uploaded_file($_FILES['arquivo']['tmp_name'], $arquivo)) {

        print "O arquivo foi gravado com sucesso.";

    } else {

        print "Erro. O arquivo não foi enviado.";

    }

}

?>

<form enctype="multipart/form-data" action="teste.php" method="POST">

    Enviar este arquivo: <input name="arquivo" type="file">

    <input type="submit" value="Envia Arquivo">

</form>

</body>

</html>

```

Capítulo 16

PHP e MySQL (Parte I)

16 - PHP E MYSQL (PARTE I)

16.1 - INTRODUÇÃO AO MYSQL

O MySQL é o gerenciador de banco de dados mais usado com o PHP. Existem muitas funções pré-definidas para manipulação de conexões com bancos de dados.

Para interagir com uma base de dados SQL existem três comandos básicos que devem ser utilizados: um que faz a conexão com o servidor de banco de dados, um que seleciona a base de dados a ser utilizada e um terceiro que executa uma “query” SQL.

16.2 - FUNÇÕES DO PHP PARA O MYSQL

16.2.1 - MYSQL_CONNECT

A função `mysql_connect` tenta uma conexão com um servidor MySQL. Deve-se passar como parâmetros: o nome do servidor (ou número IP) onde o MySQL está sendo executado, o nome de usuário e a senha deste usuário. O comando alternativo `die` trata um possível fracasso na conexão.

16.2.2 - MYSQL_SELECTDB

A função `mysql_selectdb` seleciona qual base será selecionada dentro do banco de dados que foi conectado. O comando alternativo `die` trata um possível fracasso na seleção da base, podendo ser incluída uma mensagem customizada.

16.2.3 – MYSQL_QUERY

A função `mysql_query` faz consultas à base previamente selecionada. Deve-se passar, como parâmetros, os comandos SQL apropriados. Novamente, o comando alternativo `die` pode tratar um não sucesso na consulta.

16.2.4 - EXEMPLO

```
<?php

$link = mysql_connect("127.0.0.1", "root", "") or die ("Não foi possível conectar");

mysql_select_db("banco") or die("Não foi possível selecionar o banco de dados");

$sql = "SELECT * FROM alunos";

$query = mysql_query($sql) or die("Falha na execução da consulta");

echo "Consulta executada com sucesso";

?>
```

Obs.: O comando **or die()** é executado quando a função `mysql_connect`, `mysql_select_db` ou `mysql_query` retornam um erro. Tudo que for passado por parâmetro no **die("Erro!")** será executado.

Capítulo 17

PHP e MySQLQ (Parte II)

13 – PHP e MySQL (PARTE II)

17.1 – EXIBIÇÃO (mysql_fetch_assoc)

Para que os registros da consulta sejam exibidos, deve-se usar a função **mysql_fetch_assoc**, que retorna um array com a linha atual e move para a próxima. Para imprimir todos os resultados de uma query, é necessária a utilização de uma estrutura de controle até que a função **mysql_fetch_assoc** não retorne nenhum valor (vazio).

Exemplo:

```
<?php

$sql = "SELECT * FROM alunos";

$query = mysql_query($sql) or die("Falha na execução da consulta");

while ($linha = mysql_fetch_assoc($query)) {

    echo "<b>Nome:</b>".$linha["nome"]. " <br>";

    echo "<b>E-mail:</b>".$linha["email"]. " <br>";

}

?>
```

17.2 – FUNÇÕES DO PHP PARA O MYSQL

17.2.1 – MYSQL_NUM_ROWS

Para saber a quantidade de linhas que foram retornadas de uma consulta, utiliza-se a função **mysql_num_rows**.

17.2.2 – MYSQL_ERROR

Caso alguma consulta tenha gerado um erro, utiliza-se a função `mysql_error` para visualizar o erro retornado pelo MySQL.

17.2.3 – EXEMPLO

```
<?php

$sql = "SELECT * FROM alunos";

$query = mysql_query($sql) or die("Erro: ".mysql_error());

echo "<b>Total de linhas:</b>". mysql_num_rows($query). " <br><br>";

while ($linha = mysql_fetch_assoc($query)) {

    echo "<b>Nome:</b>".$linha["nome"]. " <br>";

    echo "<b>E-mail:</b>".$linha["email"]. " <br>";

}

?>
```

Capítulo 18

PHP e MySQLQ (Parte III)

14 – PHP e MySQL (PARTE III)

18.1 – INSERÇÃO

Para inserir alguma informação no MySQL, basta passar uma query de inserção utilizando a função `mysql_query`.

Para fazer inserções dinâmicas, vamos utilizar os formulários HTML e com os valores das variáveis GET e POST montaremos a query de inserção.

Exemplo:

```
<?php

$sql = "INSERT INTO alunos SET nome = 'João', email = 'joao@gmail.com'";

$query = mysql_query($sql) or die("Erro: ".mysql_error());

?>
```

18.2 – ATUALIZAÇÃO

Para atualizar uma informação do banco de dados MySQL basta utilizar uma query de atualização utilizando a função `mysql_query` da mesma forma que é feita a inserção.

Exemplo:

```
<?php

$sql = "UPDATE alunos SET nome = 'João', email = 'joao@gmail.com' WHERE idaluno = 1";

$query = mysql_query($sql) or die("Erro: ".mysql_error());

?>
```

18.3 – EXCLUSÃO

Para excluir uma informação do banco de dados MySQL basta utilizar uma query de exclusão utilizando a função `mysql_query` da mesma forma que é feita a inserção e atualização.

Exemplo:

```
<?php
```

```
$sql = "DELETE FROM alunos WHERE idaluno = 1";
```

```
$query = mysql_query($sql) or die("Erro: ".mysql_error());
```

```
?>
```

Capítulo 19

Session

15 – SESSION

O suporte a sessões no PHP consiste em uma maneira de preservar certos dados através dos acessos subsequentes. Isto permite fazer aplicações mais personalizadas e melhorar a aparência do seu web site.

As sessões são métodos de manter (ou preservar) determinados dados a mantê-los ativos enquanto o navegador estiver aberto, ou enquanto a sessão não expirar.

19.1 – INICIANDO UMA SESSION

Ao iniciar o script PHP é preciso informar que o script poderá conter ou resgatar valores de sessão. Obrigatoriamente, esta deve ser sempre a primeira linha do arquivo .php.

Para isso utilizamos a função `session_start()`.

Exemplo:

```
<?php

    session_start();

?>
```

Exemplo:

```
<?php

    session_start();

    $sql = "SELECT * FROM usuarios WHERE login = '".$_POST["login"]."' AND
senha = '".$_POST["senha"]."'";

    $query = mysql_query($sql) or die("Erro: ".mysql_error());

    $total = mysql_num_rows($query);

    if($total == 1){

        $usuario = mysql_fetch_array($query);
```

```

        $_SESSION["login"] = $usuario["login"];

        $_SESSION["senha"] = $usuario["senha"];

    }

?>

```

19.2 – FINALIZANDO UMA SESSION

Para apagar alguns dados de sessão, você pode usar a função unset().

Exemplo:

```

<?php

    session_start();

    if(isset($_SESSION["views"])) {

        unset($_SESSION["views"]);

    }

?>

```

Para destruir completamente a sessão usa-se a função session_destroy().

Exemplo:

```

<?php

    session_start();

    if(isset($_SESSION["views"])) {

        session_destroy();

    }

?>

```

Capítulo 20

Cookies

16 – COOKIES

Cookies são formas de armazenar informações a respeito de uma sessão dentro do disco rígido do usuário. O comando `setcookie` armazena um cookie com as informações que se desejam recuperar em seguida. Quando não for declarado um tempo de vida, o cookie se auto-destrói quando a sessão é encerrada (quando o browser for fechado).

20.1 – INICIANDO UMA COOKIE

A função `setcookie()` é usado para definir um cookie.

Sintaxe:

```
setcookie(name, value, expire, path, domain);
```

name: O nome do cookie.

value: O valor do cookie.

expire: O tempo para o cookie expirar.

path: O caminho no servidor aonde o cookie estará disponível.

domain: O domínio para qual o cookie estará disponível.

Exemplo:

O exemplo abaixo cria o cookie “nome” e atribuindo o valor “Alfama Cursos” a ele. Esse mesmo cookie terá o tempo de expiração de uma hora.

```
<?php

    setcookie("nome", "Alfama Cursos", time()+3600);

    echo $_COOKIE["nome"];

    print_r($_COOKIE);

?>
```

20.2 – FINALIZANDO UMA COOKIE

Para finalizar um cookie, é preciso garantir que a data de validade esteja expirada.

Exemplo:

```
<?php  
    setcookie("nome", "", time()-3600);  
?>
```

Capítulo 21

Include, Require, Mail

17 – INCLUDE, REQUIRE, MAIL

21.1 – INCLUDE

O comando include permite a inclusão de outros arquivos .php dentro do script que está sendo executado. O caminho do arquivo que será adicionado é passado por parâmetro.

Se o arquivo não for encontrado, todo o resto do script será executado normalmente.

Sintaxe:

```
include("caminho_do_arquivo");
```

Exemplo:

```
<?php

    include("funcoes.php");

    include_once("funcoes.php");

?>
```

OBS.: include_once tem a mesma função que a include, porém se o arquivo referenciado já foi incluso na página anteriormente, a função retorna false e o arquivo não é incluído.

21.2 – REQUIRE

O comando require permite a inclusão de outros arquivos .php dentro do script que está sendo executado. O caminho do arquivo que será adicionado é passado por parâmetro.

A diferença entre include e require é que caso o arquivo que esteja sendo adicionado, usando require, não for encontrado o .php retorna um fatal erro e para a execução do script.

Sintaxe:

```
require("caminho_do_arquivo");
```

Exemplo:

```
<?php

require("funcoes.php");

require_once("funcoes.php");

?>
```

OBS.: require_once tem a mesma função que a require, porém se o arquivo referenciado já foi incluído na página anteriormente, a função retorna false e o arquivo não é incluído.

21.3 – MAIL

O PHP permite que se enviem e-mails de forma automatizada.

A função mail é a responsável pelo envio dos e-mails, se o e-mail for enviado com sucesso a função retorna true, se ocorrer algum erro no envio a função retorna false.

Sintaxe:

```
mail(para,assunto,mensagem,cabeçalhos,parâmetros)
```

para: e-mail do destinatário (obrigatório)

assunto: assunto do e-mail (obrigatório)

mensagem: mensagem do e-mail (obrigatório)

cabeçalhos: cabeçalhos do e-mail (opcional)

parâmetros: parâmetro adicionais (opcional)

Exemplo:

```
<?php

$para = "email@gmail.com";

$assunto= "Alfama Cursos";

$mensagem = "Alfama Cursos - PHP";
```

```
$de = "email@hotmail.com";  
  
$cabecalho = "From: ".$de;  
  
if(mail($para,$assunto,$mensagem,$cabecalho)) {  
  
    echo "E-mail enviado.";   
  
} else {  
  
    echo "Erro ao enviar o e-mail.";   
  
}  
  
?>
```

Capítulo 22

Funções do PHP (Parte I)

18 – FUNÇÕES DO PHP (PARTE I)

22.1 – DATE

Retorna a data e hora do servidor.

| Caracter | Descrição | Exemplo de valores retornados |
|----------|---|---------------------------------------|
| Dia | --- | --- |
| d | Dia do mês, 2 dígitos com preenchimento de zero | 01 até 31 |
| D | Uma representação textual de um dia, três letras | Seg até Dom |
| j | Dia do mês sem preenchimento de zero | 1 até 31 |
| l | A representação textual completa do dia da semana | Domingo até Sábado |
| N | Representação numérica ISO-8601 do dia da semana | 1 (para Segunda) até 7 (para Domingo) |
| S | Sufixo ordinal inglês para o dia do mês, 2 caracteres | st, nd, rd ou th. Funciona bem com j |
| w | Representação numérica do dia da semana | 0 (para domingo) até 6 (para sábado) |
| z | O dia do ano (começando do 0) | 0 até 365 |
| Semana | --- | --- |

| Caracter | Descrição | Exemplo de valores retornados |
|----------|--|--|
| W | Número do ano da semana ISO-8601, semanas começam na Segunda | 42 (the 42nd week in the year) |
| Mês | --- | --- |
| F | Uma representação completa de um mês, como Janeiro ou Março | Janeiro até Dezembro |
| m | Representação numérica de um mês, com zeros | 01 a 12 |
| M | Uma representação textual curta de um mês, três letras | Jan a Dez |
| n | Representação numérica de um mês, sem zeros | 1 a 12 |
| t | Número de dias de um dado mês | 28 até 31 |
| Year | --- | --- |
| L | Se está em um ano bissexto | 1 se está em ano bissexto, 0 caso contrário. |
| o | Número do ano ISO-8601. Este tem o mesmo valor como Y, exceto que se o número da semana ISO (W) pertence ao anterior ou próximo ano, o ano é usado ao invés. | 1999 ou 2003 |
| Y | Uma representação de ano completa, 4 | 1999 ou 2003 |

| Caracter | Descrição | Exemplo de valores retornados |
|----------|--|-------------------------------|
| | dígitos | |
| y | Uma representação do ano com dois dígitos | 99 ou 03 |
| Tempo | --- | --- |
| a | Antes/Depois de meio-dia em minúsculo | am or pm |
| A | Antes/Depois de meio-dia em maiúsculo | AM or PM |
| B | Swatch Internet time | 000 até 999 |
| g | Formato 12 horas de uma hora sem preenchimento de zero | 1 até 12 |
| G | Formato 24 horas de uma hora sem preenchimento de zero | 0 até 23 |
| h | Formato 12 horas de uma hora com zero preenchendo à esquerda | 01 até 12 |
| H | Formato 24 horas de uma hora com zero preenchendo à esquerda | 00 até 23 |
| i | Minutos com zero preenchendo à esquerda | 00 até 59 |
| s | Segundos, com zero preenchendo à esquerda | 00 até 59 |

| Caracter | Descrição | Exemplo de valores retornados |
|----------------|---|--|
| u | Milisegundos | 54321 |
| Timezone | --- | --- |
| e | Identificador de Timezone | UTC,GMT,Atlantic/Azores |
| I (capital i) | Se a data está ou não no horário de verão | 1 se horário de verão, 0 caso contrário. |
| O | Diferença para Greenwich time (GMT) em horas | +0200 |
| P | Diferença para Greenwich time (GMT) com dois pontos entre horas e minutos | +02:00 |
| T | Abreviação de Timezone | EST,MDT ... |
| Z | Fuso horário em segundos. | -43200 até 50400 |
| Full Date/Time | --- | --- |
| c | Data em ISO 8601 | 2004-02-12T15:19:21+00:00 |
| r | Formato de data RFC 2822 | Thu, 21 Dec 2000 16:01:07 +0200 |
| U | Segundos desde a Época Unix (Janeiro 1 1970 00:00:00 GMT) | |

Sintaxe:

date(“formato”);

Exemplo:

```
<?php echo date("d/m/Y"); ?>
```

22.2 – NL2BR

Insere quebras de linha HTML antes de todas newlines (\n) em uma string

Sintaxe:

```
nl2br("texto");
```

Exemplo:

```
<?php

$nome = "Alfama \n Cursos";

echo nl2br($nome);

//Retorna: Alfama <br /> Cursos

?>
```

22.3 – JOIN

Junta elementos de um array em uma string.

Sintaxe:

```
join("separador", array);
```

Exemplo:

```
<?php

$nomes = array("Maria", "João", "José", "Rosa");

echo join(" - ", $nomes);
```

```
//Retorna: Maria – João – José – Rosa
```

```
?>
```

22.4 – EXPLODE

Divide uma string, retornando um array de strings.

Sintaxe:

```
explode (“separador”, string);
```

Exemplo:

```
<?php
```

```
$nomes = “Maria – João – José – Rosa”;
```

```
echo explode(“ – ”,$nomes);
```

```
//Retorna: array(“Maria”, “João”, “José”, “Rosa”);
```

```
?>
```

Capítulo 23

Funções do PHP (Parte II)

19 – FUNÇÕES DO PHP (PARTE II)

23.1 – TRIM

Retira espaço no início e final de uma string.

Sintaxe:

```
trim("texto");
```

Exemplo:

```
<?php

$nome = " Alfama Cursos ";

echo trim($nome);

//Retorna: "Alfama Cursos";

?>
```

23.2 – STRTOLOWER

Converte uma string para minúsculas.

Sintaxe:

```
strtolower("texto");
```

Exemplo:

```
<?php

$nome = "Alfama Cursos";

echo strtolower($nome);

//Retorna: "alfama cursos";
```

?>

23.3 – STRTOUPPER

Converte uma string para maiúsculas.

Sintexe:

```
strtoupper("texto");
```

Exemplo:

```
<?php
    $nome = "Alfama Cursos";
    echo strtoupper($nome);
    //Retorna: "ALFAMA CURSOS";
?>
```

23.4 – UNSET

Destrói a variável especificada.

Sintexe:

```
unset($variavel);
```

Exemplo:

```
<?php
    $nome = "Alfama Cursos";
    $curso = "PHP";
```

```
unset($nome, $curso);
```

```
?>
```

23.5 – STR_REPLACE

Substitui todas as ocorrências da string de procura pela a string de substituição

Sintaxe:

```
str_replace(“procura”, “substituição”, string);
```

Exemplo:

```
<?php
```

```
    $curso = “Alfama Cursos - JavaScript”;
```

```
    echo str_replace(“JavaScript”, “PHP”, $curso);
```

```
    //Retorna: “Alfama Cursos - PHP”;
```

```
?>
```

22.6 - EXERCÍCIO PROPOSTO

1)

Capítulo 24

PHP na Prática

20 – PHP NA PRÁTICA

24.1 – CLASSE DE CONTATO

Criar uma classe com o nome Contato;

Criar o método adicionarContato para adicionar novos contatos na tabela contatos do banco de dados;

Criar o método removerContato para remover contatos na tabela contatos do banco de dados;

Criar o método modificarContato para modificar contatos na tabela contatos do banco de dados.

```
<?php

class Contato {

    var $idcontato;

    var $nome;

    var $telefone;

    var $email;

    function adicionarContato($nome, $telefone, $email) {

        $this->nome = $nome;

        $this->telefone = $telefone;

        $this->email = $email;

        $sql = "INSERT INTO contatos SET nome = ".$this->nome.", telefone = ".$this->telefone.", email = ".$this->email."";

        mysql_query($sql) or die("Erro:".mysql_error());

        $this->idcontato = mysql_insert_id();

        return $this->idcontato;
    }
}
```

```
}
```

```
function removerContato($idcontato) {
```

```
    $this->idcontato = $idcontato;
```

```
    $sql = "DELETE FROM contatos WHERE idcontato = ".$idcontato."";
```

```
    mysql_query($sql) or die("Erro:".mysql_error());
```

```
}
```

```
function modificarContato($idcontato, $nome, $telefone, $email) {
```

```
    $this->idcontato = $idcontato;
```

```
    $this->nome = $nome;
```

```
    $this->telefone = $telefone;
```

```
    $this->email = $email;
```

```
    $sql = "UPDATE contatos SET nome = ".$this->nome.", telefone = ".$this->telefone.", email = ".$this->email." WHERE idcontato = ".$this->idcontato."";
```

```
    mysql_query($sql) or die("Erro:".mysql_error());
```

```
    return $this->idcontato;
```

```
}
```

```
}
```

```
?>
```

24.2 – FORMULÁRIOS

Criar um formulário HTML com os campos nome e telefone e e-mail;

Adicionar um novo contato usando o método POST;

Para modificar um contato já adicionado, passar o idcontato por GET e trazer o formulário já preenchido para ser feita a modificação.

verifica.login.php:

```
<?php

session_start();

if($_POST["login"] && $_POST["senha"]) {

    $_SESSION["login"] = $_POST["login"];

    $_SESSION["senha"] = $_POST["senha"];

} elseif($_GET["login"] == "sair") {

    unset($_SESSION["login"]);

    unset($_SESSION["senha"]);

    session_destroy();

}

if(!$_SESSION["login"] && !$_SESSION["senha"]) {

    header("Location: login.php");

}

?>
```

bancodedados.php:

```
<?php

mysql_connect("127.0.0.1", "root", "") or die("Não foi possível conectar");

mysql_select_db("alfama") or die("Não foi possível selecionar o banco de dados");

?>
```

```

<?php

include("verifica.login.php");

include("bancodedados.php");

if($_GET["idcontato"]) {

    $sql = "SELECT * FROM contatos WHERE idcontato = ".$_GET["idcontato"];

    $query = mysql_query($sql);

    $linha = mysql_fetch_assoc($query);

}

?>

<html>

<head>

<title>Alfama Cursos</title>

</head>

<body>

<table cellpadding="2" cellspacing="2" border="1">

<tr>

<td>

<form action="listagem.php" method="post" enctype="multipart/form-data">

<?php if($_GET["idcontato"]) { ?>

<input type="hidden" name="op" id="op" value="modificar" />

<input type="hidden" name="idcontato" id="idcontato" value="<?php echo
$_GET["idcontato"]; ?>" />

```

```

<?php } else { ?>

    <input type="hidden" name="op" id="op" value="cadastrar" />

<?php } ?>

<table cellpadding="2" cellspacing="2" border="0">

    <tr>

        <td>Nome:</td>

        <td><input type="text" name="nome" id="nome" value="<?php echo
$linha["nome"]; ?>" /></td>

    </tr>

    <tr>

        <td>Telefone:</td>

        <td><input type="text" name="telefone" id="telefone" value="<?=
$linha["telefone"]; ?>" /></td>

    </tr>

    <tr>

        <td>E-mail:</td>

        <td><input type="text" name="email" id="email" value="<?= $linha["email"]; ?>"
/></td>

    </tr>

    <tr>

        <td><input type="submit" value="Salvar" /></td>

    </tr>

</table>

</form>

```

```

        </td>

    </tr>

</table>

</body>

</html>

```

24.3 – LISTAGEM

Tela com a listagem de todos os contatos já cadastrados no banco de dados;

Link para modificar o contato;

Link para remover o contato.

```

<?php

include("verifica.login.php");

include("bancodedados.php");

require_once("contato.class.php");

$contato = new Contato();

if($_POST["op"] == "cadastrar") {

    $contato->adicionarContato($_POST["nome"], $_POST["telefone"], $_POST["email"]);

}elseif($_POST["op"] == "modificar") {

    $contato->modificarContato($_POST["idcontato"], $_POST["nome"], $_POST["telefone"],
    $_POST["email"]);

}elseif($_GET["op"] == "remover") {

    $contato->removerContato($_GET["idcontato"]);

```

```

    }

?>

<html>

<head>

<title>Alfama Cursos</title>

</head>

<body>

<table cellpadding="2" cellspacing="2" border="1">

<tr>

<td><a href="formulario.php">Adicionar novo contato</a></td>

<td><a href="?login=sair">Sair</a></td>

<tr>

<tr>

<td colspan="2">

<table cellpadding="2" cellspacing="2" border="1">

<tr>

<td>Id</td>

<td>Nome</td>

<td>Telefone</td>

<td>E-mail</td>

<td>Opcoes</td>

</tr>

<?php

```

```

$sql = "SELECT * FROM contatos";

$query = mysql_query($sql);

while($linha = mysql_fetch_assoc($query)) {

    ?>

    <tr>

        <td><?php echo $linha["idcontato"]; ?></td>

        <td><?php echo $linha["nome"]; ?></td>

        <td><?php echo $linha["telefone"]; ?></td>

        <td><?php echo $linha["email"]; ?></td>

        <td><a href="formulario.php?idcontato=<?php echo $linha["idcontato"];
?>">Editar</a> | <a href="?op=remover&idcontato=<?php echo $linha["idcontato"];
?>">Remover</a></td>

    </tr>

    <?php } ?>

</table>

</td>

</tr>

</table>

</body>

</html>

```

24.4 – LOGIN

Criar uma tela de login com os campos login e senha;

Só será permitido o acesso a tela de contatos se o usuário logar;

Criar um link de sair (logout/ logoff).

```
<html>

<head>

<title>Alfama Cursos</title>

</head>

<body>

<table cellpadding="2" cellspacing="2" border="1">

<tr>

<td>

<form action="listagem.php" method="post" enctype="multipart/form-data">

<table cellpadding="2" cellspacing="2" border="0">

<tr>

<td>Login:</td>

<td><input type="text" name="login" id="login" /></td>

</tr>

<tr>

<td>Senha:</td>

<td><input type="password" name="senha" id="senha" /></td>

</tr>

<tr>

<td><input type="submit" value="Entrar" /></td>

</tr>

</table>

</form>

</td>
```

```
</tr>
</table>
</body>
</html>
```

REFERÊNCIAS BIBLIOGRÁFICAS

WELLING, Luke. Php e Mysql - Desenvolvimento. 3. ed. São Paulo : Campus, 2005.

EDITORA Europa. Curso de PHP. www.com.br, São Paulo, n. 40, set. 2003. CD-ROM.

MANUAL OFICIAL PHP. Capturado em 01 dezembro. 2012. Disponível na Internet
http://www.php.net/manual/pt_BR/



alfamacursos.com.br